EDB Postgres™ Backup and Recovery Guide

EDB Postgres™ Backup and Recovery 1.1.1
formerly EDB Backup and Recovery Tool 1.1.1

March 8, 2016

EDB Postgres™ Backup and Recovery Guide, Version 1.1.2
by EnterpriseDB® Corporation

EnterpriseDB Corporation, 34 Crosby Drive, Suite 100, Bedford, MA 01730, USA
T +1 781 357 3390 **F** +1 978 467 1307 **E** info@enterprisedb.com **www**.enterprisedb.com

Table of Contents

1 Introduction

Notice:

The names for EDB's products have changed. The product formerly referred to as 'EDB Backup and Recovery Tool' is now referred to as 'EDB Postgres Backup and Recovery'. Until a new version of this documentation is published, wherever you see 'EDB Backup and Recovery Tool' you may substitute it with 'EDB Postgres Backup and Recovery'. Name changes in software and software outputs will be phased in over time.

The *EDB Backup and Recovery Tool* (hereafter referred to as *BART*) is an administrative utility providing simplified backup and recovery management for multiple local or remote Postgres Plus® Advanced Server and PostgreSQL® database servers.

The following are some of the main features provided by BART:

- Supports complete, hot, physical backups of multiple Postgres Plus Advanced Server and PostgreSQL database servers
- Provides backup and recovery management of the database servers on local or remote hosts
- Uses a single, centralized catalog for backup data
- Provides retention policy support for defining and managing how long backups should be kept
- Provides the capability to store the backup data in compressed format
- Verifies backup data with checksums
- Displays backup information in an easy-to-read form
- Simplifies the point-in-time recovery process

The following chapters provide the information needed to install, configure, and use BART:

- Chapter 2 provides an overview of the BART components and concepts.
- Chapter 3 provides instructions for installing BART, upgrading from previous versions of BART, and uninstalling BART.
- Chapter 4 provides the steps for configuring BART and the database servers to be managed for backup and recovery.
- Chapter 5 describes the backup and recovery management process using BART.
- Chapter 6 provides a comprehensive example of both local and remote database server configuration and operation.
- Chapter 7 lists known issues.

The remaining sections in this chapter describe basic conventions used throughout this document.

1.1 What's New

The following features have been added to BART 1.0 to produce BART 1.1.

- The BART configuration process has been simplified with the `INIT` subcommand that performs functions such as creating the BART backup catalog, automatically configuring the `archive_command` configuration parameter (for Postgres version 9.4), and rebuilding the backupinfo file, which contains the information for each backup. For more information, see sections 4.2.3.2 and 5.4.1.
- Retention policies can now be set that determine when a backup should be considered obsolete and ready for deletion. A retention policy can be defined according to redundancy (maximum number of backups for a given database server) or by a recovery window (age in number of days, weeks, or months). A policy can be applied globally to all database servers or individually by database server. For more information, see Section 5.2.
- When using BART subcommands, backups can now be identified by a user-defined and hence, user-friendly alphanumeric name. Previously, only the integer backup identifier could be used to reference a backup. Backup names may include variables that are substituted by the year, month, day, hour, minute, or second of the backup's creation timestamp. For more information, see sections 4.2.5 and 5.4.2.
- The `SHOW-BACKUPS` subcommand now includes the `-t` option to display more comprehensive backup information in list format. Previously, only the default tabular format was displayed. For more information, see Section 5.4.4.
- The archived WAL files can now be stored in the BART backup catalog in compressed format. For more information, see sections 4.1, 4.2.5, and 5.4.6.
- The `BACKUP` subcommand now detects if there is not enough disk space available in the BART backup catalog to perform a base backup. An alert is issued before any attempt is made to copy backup files to the BART backup catalog, thus avoiding wasted time and disk space resulting from an incomplete backup. For more information, see Section 5.4.2.
- If the `BACKUP` subcommand is specified with the `-s all` option to take backups of multiple database servers, and one or more database servers are not accessible, then the `BACKUP` operation skips the inaccessible servers and continues with the backup of the other database servers. Previously, the `BACKUP` subcommand failed as soon as the first inaccessible server was encountered. For more information, see Section 5.4.2.
- The `BACKUP` subcommand now generates checksums for tablespaces. For more information, see Section 4.2.4.
- The `RESTORE` subcommand now defaults to the restoration of the latest (that is, the most recent) base backup if the `-i` option is omitted. This avoids the necessity of finding and specifying the backup identifier or name if it is simply desired to restore the most recent backup. For more information, see Section 5.4.7. In addition, the `SHOW-BACKUPS` subcommand not lists the backups in descending

order by backup date/time (that is, the most recent backups appear at the top of the list and the oldest backups are at the bottom).

- The `DELETE` subcommand now supports deletion of multiple, specified backups. Previously, only one backup or all backups could be specified. In addition, a dry run option, `-n`, can be specified so that the potential results can be displayed without actually performing the physical backup deletion. For more information, see Section 5.4.8.
- The `VERIFY-CHKSUM` subcommand now supports verification of checksums for all database servers with the `-s` option. Previously, a single, specific database server was required. In addition, the `-s` and `-i` options can now be omitted, which default to all database servers and checksums of all backups, respectively. For more information, see Section 5.4.5.
- A choice is now available to either copy the archived WAL files to a local directory where the database server is being restored before the database server archive recovery begins, or to stream the WAL files directly from the BART backup catalog during the database server archive recovery. This choice is determined by the `copy_wals_during_restore` parameter in the BART configuration file (see sections 4.1 and 4.2.5) or the `-c` option of the `RESTORE` subcommand (see Section 5.4.7).

1.2 Typographical Conventions Used in this Guide

Certain typographical conventions are used in this manual to clarify the meaning and usage of various commands, statements, programs, examples, etc. This section provides a summary of these conventions.

In the following descriptions a *term* refers to any word or group of words that are language keywords, user-supplied values, literals, etc. A term's exact meaning depends upon the context in which it is used.

- *Italic font* introduces a new term, typically, in the sentence that defines it for the first time.
- `Fixed-width (mono-spaced) font` is used for terms that must be given literally such as SQL commands, specific table and column names used in the examples, programming language keywords, etc. For example, `SELECT * FROM emp;`
- *`Italic fixed-width font`* is used for terms for which the user must substitute values in actual usage. For example, `DELETE FROM` *`table_name`*;
- A vertical pipe | denotes a choice between the terms on either side of the pipe. A vertical pipe is used to separate two or more alternative terms within square brackets (optional choices) or braces (one mandatory choice).

- Square brackets [ ] denote that one or none of the enclosed term(s) may be substituted. For example, `[ a | b ]`, means choose one of "`a`" or "`b`" or neither of the two.
- Braces {} denote that exactly one of the enclosed alternatives must be specified. For example, `{ a | b }`, means exactly one of "`a`" or "`b`" must be specified.
- Ellipses ... denote that the preceding term may be repeated. For example, `[ a | b ] ...` means that you may have the sequence, "`b a a b a`".

1.3 Other Conventions Used in this Guide

The following is a list of other conventions used throughout this document.

- Much of the information in this document applies interchangeably to the PostgreSQL and Postgres Plus Advanced Server database systems. The term *Postgres* is used to generically refer to both PostgreSQL and Postgres Plus Advanced Server. When a distinction needs to be made between these two database systems, the specific names, PostgreSQL or Advanced Server are used.
- The installation directory path of the PostgreSQL or Postgres Plus Advanced Server products is referred to as *`POSTGRES_INSTALL_HOME`*. For PostgreSQL Linux installations, this defaults to `/opt/PostgreSQL/`*`version_no`*. For PostgreSQL Windows installations, this defaults to `C:\Program Files\PostgreSQL\`*`version_no`*. For Advanced Server Linux installations, this defaults to `/opt/PostgresPlus/`*`version_no`*. For Advanced Server Windows installations, this defaults to `C:\Program Files\PostgresPlus\`*`version_no`*. The product version number is represented by *`version_no`*.

2 Overview

BART provides a simplified interface for the continuous archiving and point-in-time recovery method provided with Postgres database servers. This consists of the following processes:

- Capturing an image of a database cluster called a *base backup*, which is used as a starting point for recovery
- Archiving the *Write-Ahead Log segments* (WAL files), which continuously record changes to be made to the database files
- Performing *Point-In-Time Recovery* (PITR) to a specified transaction ID or timestamp with respect to a timeline using the base backup and the WAL files

Detailed information regarding these processes is documented in the *PostgreSQL Core Documentation* available at:

http://www.postgresql.org/docs/9.5/static/continuous-archiving.html

BART utilizes the PostgreSQL `pg_basebackup` utility program as the underlying mechanism for creating base backups.

For information about `pg_basebackup`, see the *PostgreSQL Core Documentation* available at:

http://www.postgresql.org/docs/9.5/static/app-pgbasebackup.html

These features provide a complete backup and recovery methodology for Postgres database servers, however, the management of this process can be quite complex, especially when dealing with multiple database servers in a distributed environment.

BART simplifies this management process by use of a centralized backup catalog, a single configuration file, and a command line interface controlling the necessary operations.

Reasonable defaults are automatically used for various backup and restore options. BART also performs the necessary recovery file configuration required for point-in-time recovery by means of its command line interface.

BART also provides other features to enhance backup management such as the following:

- Automation of the WAL archiving command configuration (for Postgres version 9.4 and beyond)
- Use of retention policies to evaluate, categorize, and delete backups that are old and therefore considered obsolete

- Compression of WAL files to conserve disk space
- Customizable naming of backups to ease their usage
- Easy access to comprehensive information about each backup

The key components for using BART are the following:

- **BART Host.** The host system on which BART is installed. The BART operations are invoked from this host system and the database server base backups and archived WAL files are stored on this host as well.
- **BART User Account.** Linux operating system user account you choose to run BART. The BART user account owns the BART backup catalog directory.
- **BART Configuration File.** File in editable text format containing the configuration information used by BART.
- **BART Backup Catalog.** File system directory structure containing all of the base backups and archived WAL files for the database servers managed by BART.
- **BART Backupinfo File.** File in text format containing information for a BART backup. A backupinfo file resides in each backup subdirectory within the BART backup catalog.
- **BART Command Line Utility Program.** Single, executable file named `bart`, which is used to commence all BART operations.

Other concepts and terms referred to in this document include the following:

- **Postgres Database Cluster.** Also commonly called the *data directory*, this is the file system directory where all of the data files related to a particular Postgres database server instance are stored. (A particular, running instance is identified by its host and port number when connecting to a database.) The database cluster is identified by the `-D` option when it is created, started, stopped, etc. by the Postgres `initdb` and `pg_ctl` commands. Typically by default, the initial database cluster is located in directory *`POSTGRES_INSTALL_HOME`*`/data`. A base backup is a copy of a database cluster. **Note:** The terms database cluster and database server are used somewhat interchangeably throughout this document, though a single database server can run multiple database clusters.
- **Postgres User Account.** Operating system user account that runs the Advanced Server or PostgreSQL database server. By default, the Postgres user account is `enterprisedb` for Advanced Server installed in Oracle compatible mode. By default, the Postgres user account is `postgres` for Advanced Server installed in PostgreSQL compatible mode. For a PostgreSQL database server, this user account is also typically `postgres`.
- **Replication Database User.** For each database server managed by BART, a database superuser, or a database user with `replication` privilege must be selected to act as the replication database user. This database user is used to connect to the database server when BART invokes `pg_basebackup`. The database superusers created with an initial Postgres database server installation (`enterprisedb` or `postgres`) may be used for this purpose.

- **Secure Shell (SSH)/Secure Copy (SCP).** Linux utility programs used to log into hosts (SSH) and copy files (SCP) between hosts. A valid user account must be specified that exists on the target host and in effect, is the user account under which the SSH or SCP operations occur.

Chapter 4 provides information on how all of these components are configured and used with BART.

2.1 Prerequisites

This section describes the supported database server versions, the required supporting software, etc.

2.1.1 Supported Platforms and Database Versions

BART can be installed on the following platforms:

- CentOS 6.x or 7.x
- Red Hat Enterprise Linux (RHEL) 6.x or 7.x

Note: BART currently runs on only 64-bit platforms.

The Postgres database versions that can be managed by BART are the following:

- Postgres Plus Advanced Server version 9.1 or later
- PostgreSQL version 9.1 or later

2.1.2 Required Software

The following components must be installed on the BART host:

- PostgreSQL `libpq` library
- PostgreSQL `pg_basebackup` utility program

The BART host components can be installed in a number of different ways using EnterpriseDB packages. See Section 3.1 for installation instructions for these components.

For most scenarios using BART, the Secure Shell (SSH) server daemon must be enabled and active on the BART host as well as on any remote database server hosts on which BART will be managing backup and recovery.

The SSH and SCP client programs must also be available on the BART host as well as on the remote database server hosts.

The only case where SSH and SCP is not required is when all of the following are true:

- BART and the database server managed by BART are running on the same host. In other words, they are all local relative to each other.
- The BART user account (and therefore account owner of the BART backup catalog) is the same operating system user account running the database server. This same user account is also the owner of the database clusters and intended directories to which the database clusters are to be restored.

For the BART/database server pair described by the preceding conditions, SSH and SCP usage is not necessary. An example of this case is given in Chapter 6.

3 Installing, Upgrading, and Uninstalling BART

BART is supplied as an RPM package. Use the Yum package manager to install BART.

For information about using Yum, see the Yum project website at:

http://yum.baseurl.org/

The `edb-bart` package is located in the EnterpriseDB Tools portion of the *EDB Yum Repository*.

Please contact your EnterpriseDB Account Manager or mailto:sales@enterprisedb.com to request the location and credentials for the EDB Yum Repository.

For information about using the EDB Yum Repository see Chapter 3 of the *EDB Postgres Advanced Server Installation Guide* available from the EnterpriseDB website at:

http://www.enterprisedb.com/products-services-training/products/documentation/enterpriseedition

If you are installing BART for the first time, see Section 3.1.

If you are upgrading from BART version 1.1.0 to 1.1.1, see Section 3.2.

If you are upgrading from BART version 1.0.x, see Section 3.3.

If you wish to uninstall BART, see Section 3.4.

3.1 First-Time Installation of BART

Before installing BART, you must have the `libpq` library installed on your host. In addition, you will need the `pg_basebackup` utility program.

The following are the steps to perform the installation of these components and the BART product.

Step 1: Download the Postgres Plus Advanced Server 9.4 repository configuration package (`ppas94-repo-9.4-1.noarch.rpm`) and the EnterpriseDB Tools repository configuration package (`enterprisedb-tools-repo-1.0-1.noarch.rpm`).

As the `root` user, issue the following commands to install these repository configuration packages:

```
[root@localhost ~]# rpm -ivh enterprisedb-tools-repo-1.0-1.noarch.rpm
warning: enterprisedb-tools-repo-1.0-1.noarch.rpm: Header V4 RSA/SHA1 Signature, key ID
7e30651c: NOKEY
Preparing...                ########################################### [100%]
   1:enterprisedb-tools-repo########################################### [100%]
[root@localhost ~]# rpm -ivh ppas94-repo-9.4-1.noarch.rpm
Preparing...                ########################################### [100%]
   1:ppas94-repo            ########################################### [100%]
```

Step 2: In the `/etc/yum.repos.d` directory, the repository configuration files `ppas94.repo` and `enterprisedb-tools.repo` are installed.

Edit these files by substituting your user name and password obtained from your EnterpriseDB Account Manager for the `<username>:<password>` placeholders of the `baseurl` parameter in these two files.

For example, in the `ppas94.repo` file, this parameter is the following:

```
baseurl=http://<username>:<password>@yum.enterprisedb.com/9.4/redhat/rhel-$releasever-
$basearch
```

Step 3: Install the components required by BART, which are the `libpq` library and the `pg_basebackup` utility program.

The installation of these components can be accomplished in a couple of different ways depending upon the software components you already have installed or plan to install on the BART host.

If you do not have the complete Advanced Server database server product installed on the BART host, and you do not intend to install it, the simplest approach is to install the `ppas94-server-client` package. This package contains the necessary utilities, libraries, and client programs.

If you already have the complete Advanced Server database server product installed on the BART host, or you intend to install it, see Step 4 for other alternatives.

To install the `ppas94-server-client` package, invoke the `yum install ppas94-server-client` command:

```
[root@localhost ~]# yum install ppas94-server-client
Loaded plugins: fastestmirror, refresh-packagekit, security
Loading mirror speeds from cached hostfile
 * base: mirror.cc.columbia.edu
 * extras: mirror.metrocast.net
 * updates: mirror.es.its.nyu.edu
Setting up Install Process
Resolving Dependencies
--> Running transaction check
---> Package ppas94-server-client.x86_64 0:9.4.4.9-1.rhel6 will be installed
--> Processing Dependency: ppas94-server-libs = 9.4.4.9-1.rhel6 for package: ppas94-
server-client-9.4.4.9-1.rhel6.x86_64
--> Processing Dependency: libpq.so.5()(64bit) for package: ppas94-server-client-
9.4.4.9-1.rhel6.x86_64
--> Running transaction check
---> Package ppas94-server-libs.x86_64 0:9.4.4.9-1.rhel6 will be installed
```

```
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
================================================
 Package                              Arch                     Version
Repository                Size
================================================================================
================================================
Installing:
 ppas94-server-client                 x86_64                   9.4.4.9-1.rhel6
ppas94                    861 k
Installing for dependencies:
 ppas94-server-libs                   x86_64                   9.4.4.9-1.rhel6
ppas94                    434 k

Transaction Summary
================================================================================
================================================
Install       2 Package(s)

Total download size: 1.3 M
Installed size: 6.2 M
Is this ok [y/N]: y
Downloading Packages:
(1/2): ppas94-server-client-9.4.4.9-1.rhel6.x86 64.rpm
| 861 kB     00:00
(2/2): ppas94-server-libs-9.4.4.9-1.rhel6.x86_64.rpm
| 434 kB     00:00
--------------------------------------------------------------------------------
------------------------------------------------
Total
2.2 MB/s | 1.3 MB     00:00
Running rpm_check_debug
Running Transaction Test
Transaction Test Succeeded
Running Transaction
  Installing : ppas94-server-libs-9.4.4.9-1.rhel6.x86 64
1/2
  Installing : ppas94-server-client-9.4.4.9-1.rhel6.x86_64
2/2
  Verifying  : ppas94-server-libs-9.4.4.9-1.rhel6.x86_64
1/2
  Verifying  : ppas94-server-client-9.4.4.9-1.rhel6.x86 64
2/2

Installed:
  ppas94-server-client.x86_64 0:9.4.4.9-1.rhel6

Dependency Installed:
  ppas94-server-libs.x86 64 0:9.4.4.9-1.rhel6

Complete!
```

Upon completion, you will find the `pg_basebackup` utility program located in directory `/usr/ppas-9.4/bin`. The `libpq` library is located in directory `/usr/ppas-9.4/lib`.

Make note of these locations as they are referenced later in this document.

Note: The term *`POSTGRES_INSTALL_HOME`* when referring to the BART host is the parent directory location where `pg_basebackup` and `libpq` have been installed. If you are using the `ppas94-server-client` package, or you have installed the complete `ppas94` package, *`POSTGRES_INSTALL_HOME`* is directory location `/usr/ppas-9.4`. If

you are using the Advanced Server product installed from the Postgres interactive, graphical user interface installer, the default *POSTGRES_INSTALL_HOME* directory is `/opt/PostgreSQL/9.4` if installed in PostgreSQL compatible mode or `/opt/PostgresPlus/9.4AS` if installed in Oracle compatible mode. (The latter is the directory location used in examples throughout this document.)

After you have installed the `ppas94-server-client` package, proceed to Step 5 to install BART.

Step 4: This step is for circumstances where you already have the complete Advanced Server database server product installed on your BART host, or you intend to do so.

First, you may need to install the `ppas94-server-libs` package. This is dependent upon what other Advanced Server RPM packages you have installed or plan to install.

If you have installed the complete Advanced Server product from the `ppas94` RPM package, or plan to do so, you do not need to install the `ppas94-server-libs` package. Proceed to Step 5 to install the BART product.

The following example shows the installation of the complete `ppas94` RPM package invoked by the `yum install ppas94` command:

```
[root@localhost ~]# yum install ppas94
Loaded plugins: fastestmirror, refresh-packagekit, security
Loading mirror speeds from cached hostfile
 * base: mirror.nexcess.net
 * extras: centos.mirror.constant.com
 * updates: centos.mirrors.wvstateu.edu
enterprisedb-tools
| 2.4 kB     00:00
enterprisedb-tools/primary_db
|  12 kB     00:00
ppas94
| 2.5 kB     00:00
ppas94/primary_db
|  29 kB     00:00
Setting up Install Process
Resolving Dependencies
--> Running transaction check
---> Package ppas94.x86 64 0:9.4-1.rhel6 will be installed
    .
    .
    .
```

If you have already installed, or you plan to install Postgres using the interactive, graphical user interface installer (see Chapter 4 of the *EDB Postgres Advanced Server Installation Guide* available from the EnterpriseDB website at:

http://www.enterprisedb.com/products-services-training/products/documentation/enterpriseedition),

then you must also install the `ppas94-server-libs` package by issuing the `yum install ppas94-server-libs` command as follows:

```
[root@localhost ~]# yum install ppas94-server-libs
Loaded plugins: fastestmirror, refresh-packagekit, security
Loading mirror speeds from cached hostfile
 * base: mirror.us.leaseweb.net
 * extras: mirror.fdcservers.net
 * updates: centos.mirror.constant.com
Setting up Install Process
Resolving Dependencies
--> Running transaction check
---> Package ppas94-server-libs.x86 64 0:9.4.2.7-1.rhel6 will be installed
--> Processing Dependency: libmemcached.so.2(libmemcached_2)(64bit) for package:
ppas94-server-libs-9.4.2.7-1.rhel6.x86_64
--> Processing Dependency: libmemcached.so.2()(64bit) for package: ppas94-server-libs-
9.4.2.7-1.rhel6.x86 64
--> Running transaction check
---> Package libmemcached.x86 64 0:0.31-1.1.el6 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
============================
 Package                        Arch                Version
Repository            Size
================================================================================
============================
Installing:
 ppas94-server-libs             x86_64              9.4.2.7-1.rhel6
ppas94               434 k
Installing for dependencies:
 libmemcached                   x86_64              0.31-1.1.el6
base                  80 k

Transaction Summary
================================================================================
============================
Install       2 Package(s)

Total download size: 513 k
Installed size: 1.7 M
Is this ok [y/N]: y
Downloading Packages:
(1/2): libmemcached-0.31-1.1.el6.x86_64.rpm
|  80 kB     00:00
(2/2): ppas94-server-libs-9.4.2.7-1.rhel6.x86 64.rpm
| 434 kB     00:00
--------------------------------------------------------------------------------
----------------------------
Total                                                                      648
kB/s | 513 kB     00:00
Running rpm_check_debug
Running Transaction Test
Transaction Test Succeeded
Running Transaction
Warning: RPMDB altered outside of yum.
  Installing : libmemcached-0.31-1.1.el6.x86 64
1/2
  Installing : ppas94-server-libs-9.4.2.7-1.rhel6.x86_64
2/2
  Verifying  : ppas94-server-libs-9.4.2.7-1.rhel6.x86_64
1/2
  Verifying  : libmemcached-0.31-1.1.el6.x86 64
2/2

Installed:
  ppas94-server-libs.x86_64 0:9.4.2.7-1.rhel6

Dependency Installed:
  libmemcached.x86_64 0:0.31-1.1.el6

Complete!
```

Step 5: Install the BART RPM package. This can be done by using either the `yum` command or the `rpm` command.

The following shows the installation of the current, available version of BART from the EnterpriseDB Tools repository using the `yum install edb-bart` command:

```
[root@localhost ~]# yum install edb-bart
Loaded plugins: fastestmirror, refresh-packagekit, security
Loading mirror speeds from cached hostfile
 * base: mirror.cc.columbia.edu
 * extras: mirror.metrocast.net
 * updates: mirror.es.its.nyu.edu
Setting up Install Process
Resolving Dependencies
--> Running transaction check
---> Package edb-bart.x86_64 0:1.1.0-1.rhel6 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
================================================
 Package                    Arch                     Version
Repository                              Size
================================================================================
================================================
Installing:
 edb-bart                   x86_64                   1.1.0-1.rhel6
enterprisedb-tools                     148 k

Transaction Summary
================================================================================
================================================
Install       1 Package(s)

Total download size: 148 k
Installed size: 375 k
Is this ok [y/N]: y
Downloading Packages:
edb-bart-1.1.0-1.rhel6.x86_64.rpm
| 148 kB     00:00
Running rpm_check_debug
Running Transaction Test
Transaction Test Succeeded
Running Transaction
  Installing : edb-bart-1.1.0-1.rhel6.x86_64
1/1
  Verifying  : edb-bart-1.1.0-1.rhel6.x86_64
1/1

Installed:
  edb-bart.x86_64 0:1.1.0-1.rhel6

Complete!
```

The following shows the installation of BART from a downloaded BART RPM package file using the `rpm -ivh` command:

```
[root@localhost ~]# rpm -ivh edb-bart-1.1.0-1.rhel6.x86_64.rpm
Preparing...                ########################################### [100%]
   1:edb-bart               ########################################### [100%]
```

The BART product is installed in the following directory location referred to as *BART_HOME*:

```
/usr/edb-bart-x.x
```

The following files are included in the installation:

Table 3-1 - Post-Installation Files

File Name	Location	Description
bart	*BART_HOME*/bin	BART command line, executable program
bart.cfg	*BART_HOME*/etc	BART configuration file
bart_license.txt	*BART_HOME*	License agreement

Note: The BART version number is represented by *x.x* (for example, 1.1).

You can also verify the installation by invoking a BART subcommand:

```
[root@localhost edb-bart-1.1]# ls
bart_license.txt  bin  etc
[root@localhost edb-bart-1.1]# cd bin
[root@localhost bin]# ./bart --version
bart (EnterpriseDB) 1.1.1
```

3.2 Upgrading from BART 1.1.0 to BART 1.1.1

If you are already using BART 1.1.0 and are upgrading to BART 1.1.1, as the root user, invoke the yum update edb-bart command. See Step 2 of Section 3.3 for an example of invoking the yum update edb-bart command.

BART 1.1.1 is installed in the same *BART_HOME* parent directory as BART 1.1.0.

Your previous BART configuration file you were using for BART 1.1.0 remains in the *BART_HOME*/etc directory as bart.cfg.

The newly installed BART configuration file for BART 1.1.1 is located in the same directory, but named bart.cfg.rpmnew.

If you so desire, add the new BART configuration parameters contained in the bart.cfg.rpmnew file to your existing BART configuration file, bart.cfg, or alternatively, update the bart.cfg.rpmnew file with you current BART configuration parameter settings. See Step 3 of Section 3.3 for some specific BART configuration parameters to update in order to use the same BART backup catalog.

The final BART configuration file with the parameter settings you wish to use must be named bart.cfg within the *BART_HOME*/etc directory unless you use the -c general option with the BART subcommands as described in Section 5.3.

After you have completed your update to the BART configuration file, you are ready to use BART 1.1.1.

3.3 Upgrading from BART 1.0.x

If you are already using BART 1.0.x (that is, BART 1.0 or BART 1.0.2) perform the following steps to upgrade to BART 1.1:

Step 1: It is suggested that you make a copy of your BART configuration file located in the *BART_HOME*/etc directory although installation of BART 1.1 results in a new, separate *BART_HOME* parent directory.

Step 2: As the root user, update to BART 1.1 with the yum update edb-bart command.

The following is an example of an update from an older BART version to a newer one:

```
[root@localhost ~]# yum update edb-bart
Loaded plugins: fastestmirror, refresh-packagekit, security
Loading mirror speeds from cached hostfile
 * base: mirror.cc.columbia.edu
 * extras: mirror.metrocast.net
 * updates: mirror.es.its.nyu.edu
Setting up Update Process
Resolving Dependencies
--> Running transaction check
---> Package edb-bart.x86_64 0:1.0.2-1.rhel6 will be updated
---> Package edb-bart.x86_64 0:1.1.0-1.rhel6 will be an update
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
================================================================
 Package                     Arch                       Version
Repository                                Size
================================================================================
================================================================
Updating:
 edb-bart                    x86_64                     1.1.0-1.rhel6
enterprisedb-tools                     148 k

Transaction Summary
================================================================================
================================================================
Upgrade       1 Package(s)

Total download size: 148 k
Is this ok [y/N]: y
Downloading Packages:
edb-bart-1.1.0-1.rhel6.x86_64.rpm
| 148 kB     00:00
Running rpm_check_debug
Running Transaction Test
Transaction Test Succeeded
Running Transaction
```

```
  Updating   : edb-bart-1.1.0-1.rhel6.x86_64
1/2
  Cleanup    : edb-bart-1.0.2-1.rhel6.x86_64
2/2
  Verifying  : edb-bart-1.1.0-1.rhel6.x86_64
1/2
  Verifying  : edb-bart-1.0.2-1.rhel6.x86_64
2/2

Updated:
  edb-bart.x86_64 0:1.1.0-1.rhel6

Complete!
```

Step 3: Update the new BART configuration file located in the new *BART_HOME*/etc directory with the configuration file settings you wish to retain from your previous BART version that you saved in Step 1.

If you have a BART backup catalog created with BART 1.0.x that you wish to use with BART 1.1, be sure you set the backup_path parameter of the BART 1.1 configuration file to this existing BART 1.0.x backup catalog.

The bart-host parameter must also specify the same BART user account that was used for BART 1.0.x to be able to access the BART backup catalog.

Step 4: In order to use the BART backups created with BART 1.0.x, you must first run the BART 1.1 INIT subcommand with the -r option in order to create a BART backupinfo file for each backup.

The BART backupinfo file is a new component introduced with BART 1.1 that stores the information about each backup. An upgrade example is shown by the following.

The content of an existing BART 1.0.x backup catalog is shown by the following:

```
[root@localhost ppas93]# pwd
/opt/backup/ppas93
[root@localhost ppas93]# ls -l
total 24
drwx------ 2 enterprisedb enterprisedb 4096 Apr  3 12:19 1428077978301
drwx------ 2 enterprisedb enterprisedb 4096 Apr  3 12:22 1428078124328
drwx------ 2 enterprisedb enterprisedb 4096 Apr  3 12:23 1428078196662
drwx------ 2 enterprisedb enterprisedb 4096 Apr  3 12:25 1428078299553
drwxrwxr-x 3 enterprisedb enterprisedb 4096 Apr  3 12:28 1428078508043
drwx------ 2 enterprisedb enterprisedb 4096 Apr  3 12:38 archived_wals
root@localhost ppas93]# ls -l 1428077978301
total 54652
-rw-rw-r-- 1 enterprisedb enterprisedb       33 Apr  3 12:19 base.md5
-rw-rw-r-- 1 enterprisedb enterprisedb 55954432 Apr  3 12:19 base.tar
[root@localhost ppas93]# ls -l 1428078124328
total 54556
-rw-rw-r-- 1 enterprisedb enterprisedb       33 Apr  3 12:22 base.md5
-rw-rw-r-- 1 enterprisedb enterprisedb 55854080 Apr  3 12:22 base.tar
[root@localhost ppas93]# ls -l 1428078196662
total 54716
-rw-rw-r-- 1 enterprisedb enterprisedb       33 Apr  3 12:23 base.md5
```

```
-rw-rw-r-- 1 enterprisedb enterprisedb 56023040 Apr  3 12:23 base.tar
[root@localhost ppas93]# ls -l 1428078299553
total 5516
-rw-rw-r-- 1 enterprisedb enterprisedb      33 Apr  3 12:25 base.md5
-rw-rw-r-- 1 enterprisedb enterprisedb 5643210 Apr  3 12:25 base.tar.gz
[root@localhost ppas93]# ls -l 1428078508043
total 4
drwx------ 17 enterprisedb enterprisedb 4096 Apr  3 12:28 base
```

After the installation and setup of BART 1.1 is completed, run the BART 1.1 `INIT` subcommand with the `-r` option. See Section 5.4.1 for additional information regarding the `INIT` subcommand.

Note: The `LD_LIBRARY_PATH` environment variable must be set before running BART 1.1 as shown by the following example. See Section 5.3 for additional information.

```
-bash-4.1$ export LD_LIBRARY_PATH=/opt/PostgresPlus/9.4AS/lib:$PATH
-bash-4.1$ ./bart INIT -r
INFO:  rebuilding BACKUPINFO for backup '1428078508043' of server 'ppas93'
INFO:  rebuilding BACKUPINFO for backup '1428078299553' of server 'ppas93'
INFO:  backup checksum: 8467b3d82c88ae4a9c700c5b37eb5977 of base.tar.gz
INFO:  backup checksum: a40463135e6fcd156a098c3e3e800c46 of base.md5
INFO:  rebuilding BACKUPINFO for backup '1428078196662' of server 'ppas93'
INFO:  backup checksum: 8bc156c7bb3891ac5a3eb37a1e201503 of base.md5
INFO:  backup checksum: d4f182880f74282697237c0543ec8817 of base.tar
INFO:  rebuilding BACKUPINFO for backup '1428078124328' of server 'ppas93'
INFO:  backup checksum: e58ea5c2e254834a3a412dcccb5ff5a3 of base.md5
INFO:  backup checksum: 7971e77c62132bcfe169aa34ffc9b8e3 of base.tar
INFO:  rebuilding BACKUPINFO for backup '1428077978301' of server 'ppas93'
INFO:  backup checksum: d9b679e87032de7da197ae18496004b7 of base.md5
INFO:  backup checksum: 84b010b0312057321050e0482ef62f07 of base.tar
```

A backupinfo file is now located in each backup subdirectory as shown by the following:

```
[root@localhost ppas93]# pwd
/opt/backup/ppas93
[root@localhost ppas93]# ls -l 1428077978301
total 54656
-rw-rw-r-- 1 enterprisedb enterprisedb      795 Apr  3 12:48 backupinfo
-rw-rw-r-- 1 enterprisedb enterprisedb       33 Apr  3 12:19 base.md5
-rw-rw-r-- 1 enterprisedb enterprisedb 55954432 Apr  3 12:19 base.tar
[root@localhost ppas93]# ls -l 1428078124328
total 54560
-rw-rw-r-- 1 enterprisedb enterprisedb      795 Apr  3 12:48 backupinfo
-rw-rw-r-- 1 enterprisedb enterprisedb       33 Apr  3 12:22 base.md5
-rw-rw-r-- 1 enterprisedb enterprisedb 55854080 Apr  3 12:22 base.tar
[root@localhost ppas93]# ls -l 1428078196662
total 54720
-rw-rw-r-- 1 enterprisedb enterprisedb      795 Apr  3 12:48 backupinfo
-rw-rw-r-- 1 enterprisedb enterprisedb       33 Apr  3 12:23 base.md5
-rw-rw-r-- 1 enterprisedb enterprisedb 56023040 Apr  3 12:23 base.tar
[root@localhost ppas93]# ls -l 1428078299553
total 5520
-rw-rw-r-- 1 enterprisedb enterprisedb     806 Apr  3 12:48 backupinfo
-rw-rw-r-- 1 enterprisedb enterprisedb      33 Apr  3 12:25 base.md5
-rw-rw-r-- 1 enterprisedb enterprisedb 5643210 Apr  3 12:25 base.tar.gz
[root@localhost ppas93]# ls -l 1428078508043
total 8
-rw-rw-r--  1 enterprisedb enterprisedb  608 Apr  3 12:48 backupinfo
```

```
drwx------ 17 enterprisedb enterprisedb 4096 Apr  3 12:28 base
```

The BART 1.1 `SHOW-BACKUPS` subcommand displays the backups:

```
-bash-4.1$ ./bart SHOW-BACKUPS
 SERVER NAME   BACKUP ID       BACKUP TIME               BACKUP SIZE   WAL(s) SIZE
WAL FILES   STATUS

 ppas93        1428078508043   2015-04-03 12:28:28 EDT   52.75 MB      128.00 MB     8
active
 ppas93        1428078299553   2015-04-03 12:25:03 EDT   5.38 MB       48.00 MB      3
active
 ppas93        1428078196662   2015-04-03 12:23:19 EDT   53.43 MB      32.00 MB      2
active
 ppas93        1428078124328   2015-04-03 12:22:09 EDT   53.27 MB      48.00 MB      3
active
 ppas93        1428077978301   2015-04-03 12:19:38 EDT   53.36 MB      48.00 MB      3
active
```

You will now be able to use all of the features of BART 1.1 on the backups.

3.4 Uninstallation of BART

Decide if you want to permanently save or just delete the backup files and archived WAL files in the BART backup catalog. Uninstalling BART does not affect these files. You can delete all of the files with the BART `DELETE` subcommand (see Section 5.4.8), or use the Linux command `rm -rf /opt/backup` to delete the BART backup catalog `/opt/backup`, for example.

To uninstall BART, as the `root` user invoke the `yum remove edb-bart` command.

The following is an example of the removal of BART:

```
[root@localhost ~]# yum remove edb-bart
Loaded plugins: fastestmirror, refresh-packagekit, security
Setting up Remove Process
Resolving Dependencies
--> Running transaction check
---> Package edb-bart.x86_64 0:1.1.0-1.rhel6 will be erased
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
================================================
 Package                    Arch                    Version
Repository                              Size
================================================================================
================================================
Removing:
 edb-bart                   x86_64                  1.1.0-1.rhel6
@enterprisedb-tools                    375 k

Transaction Summary
================================================================================
================================================
Remove        1 Package(s)
```

```
Installed size: 375 k
Is this ok [y/N]: y
Downloading Packages:
Running rpm_check_debug
Running Transaction Test
Transaction Test Succeeded
Running Transaction
  Erasing    : edb-bart-1.1.0-1.rhel6.x86_64
1/1
  Verifying  : edb-bart-1.1.0-1.rhel6.x86_64
1/1

Removed:
  edb-bart.x86_64 0:1.1.0-1.rhel6

Complete!
```

4 Configuration

The main configuration steps of BART are the following:

- Establish a BART user account.
- Create the initial settings for the BART configuration file.
- Create the BART backup catalog.
- Perform the configuration setup for each database server that is to be managed by BART. The database server setup process is described in Section 4.2.

The following section describes the initial BART configuration steps.

4.1 Configuring the BART Host

This section describes the basic setup steps on the BART host. As you perform these steps, there are parameters in the BART configuration file (*BART_HOME*/etc/bart.cfg) that correspond to these settings. Edit these parameters in the BART configuration file as you proceed through the steps.

The following is an example of the minimal, required parameters in the BART configuration file that must be explicitly set:

```
[BART]
bart-host = bartuser@192.168.2.22
backup_path = /opt/backup
pg_basebackup_path = /opt/PostgresPlus/9.4AS/bin/pg_basebackup
```

For each database server, the minimal BART parameter settings are as follows:

```
[PG94]
host = 192.168.2.24
port = 5432
user = postgres
```

Note: The `port` parameter setting is only required if the database server listens on a port other than 5444.

All other parameters use default actions.

The sections in this chapter provide explicit details on the meaning and usage of the BART configuration file parameters.

The following are the parameters in the BART configuration file that apply globally to BART backup and recovery management (that is, applies to all BART managed database servers as well).

- **[BART].** Identifies the global section of the configuration file. This parameter is required and must be named BART.
- **bart-host.** IP address of the host on which BART is installed. The value for this parameter must be specified in the form *bart_user*@*bart_host_address* where *bart_user* is the operating system user account on the BART host that is used to run BART and owns the BART backup catalog directory. *bart_host_address* is the IP address of the BART host. This parameter is required.
- **backup_path.** Specifies the file system parent directory where all BART database server base backups and archived WAL files are stored. This parameter is required.
- **pg_basebackup_path.** Specifies the path to the `pg_basebackup` program that you installed on the BART host. This parameter is required.
- **xlog-method.** Determines how the transaction log is collected during execution of `pg_basebackup` through the `BACKUP` subcommand. Set to `fetch` to collect the transaction log files after the backup has completed. Set to `stream` to stream the transaction log in parallel with the base backup creation. If `stream` is used, the `max_wal_senders` configuration parameter in the `postgresql.conf` file for affected database servers must account for an additional session for the streaming of the transaction log, (that is, the setting must be a minimum of `2`). The setting of the `xlog-method` parameter in the server section of the BART configuration file overrides the setting of `xlog-method` in the global section for that particular database server. If omitted in the server section, the setting of `xlog-method` in the global section is used. If the `xlog-method` parameter is not specified in either section, the default is `fetch`. **Note:** The `xlog-method = stream` setting must not be applied to any database server containing user-defined tablespaces. See Section 4.2.4 for additional information.
- **retention_policy.** Determines when an active backup should be marked as obsolete, and hence, be a candidate for deletion. The setting can be either *max_number* `BACKUPS`, *max_number* `DAYS`, *max_number* `WEEKS`, or *max_number* `MONTHS` where *max_number* is a positive integer. If all of the keywords `BACKUPS`, `DAYS`, `WEEKS`, and `MONTHS` are omitted, the specified integer is interpreted as *max_number* `BACKUPS` by default. The setting of the `retention_policy` parameter in the server section of the BART configuration file overrides the setting of `retention_policy` in the global section for that particular database server. If omitted in the server section, the setting of `retention_policy` in the global section is used. If the `retention_policy` parameter is not specified in either section, then no additional backups are marked as obsolete when the `MANAGE` subcommand is used. See Section 5.2 for information on managing backups using a retention policy.
- **wal_compression.** Enables the compression of archived WAL files in the BART backup catalog when the `MANAGE` subcommand is invoked. Set to `enabled` to compress the archived WALS files in gzip format. Set to `disabled` to leave the files uncompressed. **Note:** The gzip compression program must be in the BART user account's `PATH`. The setting of the `wal_compression` parameter in the

server section of the BART configuration file overrides the setting of `wal_compression` in the global section for that particular database server. If omitted in the server section, the setting of `wal_compression` in the global section is used. If the `wal_compression` parameter is not specified in either section, the default is `disabled`. See Section 5.4.6 for information on using the `MANAGE` subcommand for WAL compression.

- **copy_wals_during_restore.** Determines how the archived WAL files are collected when invoking the `RESTORE` subcommand. Set to `enabled` to copy the archived WAL files from the BART backup catalog to the *`restore_path`*`/archived_wals` directory prior to the database server archive recovery. Set to `disabled` to retrieve the archived WAL files directly from the BART backup catalog during the database server archive recovery. The BART generated `restore_command` parameter in the `recovery.conf` file reflects which of the two options is used. If the `RESTORE` subcommand is invoked with the `-c` option, then the archived WAL files are copied from the BART backup catalog to the *`restore_path`*`/archived_wals` directory, thus overriding any setting of the `copy_wals_during_restore` parameter in the BART configuration file. If the `RESTORE` subcommand is invoked without the `-c` option, then the following determines how the archived WAL files are retrieved: An explicit setting of the `copy_wals_during_restore` parameter in the server section of the BART configuration file overrides the setting of `copy_wals_during_restore` in the global section for that particular database server. If omitted in the server section, the setting of `copy_wals_during_restore` in the global section is used. If the `copy_wals_during_restore` parameter is not explicitly set in either section, the default is `disabled`. See Section 5.4.7 for additional information.
- **logfile.** Specifies the path to the BART log file. This parameter is optional. If no path to a log file is specified after `logfile =`, or if the parameter is commented out, BART does not create a log file. **Note:** During the course of BART usage, any error message you may receive regarding the BART log file is typically caused by the presence of an existing log file that is not owned by the current BART user account. To resolve the problem, delete or rename the existing log file and allow BART to create a new one owned by the current BART user account.

The following is an example of the global section:

```
[BART]
bart-host = bartuser@192.168.2.22
backup_path = /opt/backup
pg_basebackup_path = /opt/PostgresPlus/9.4AS/bin/pg_basebackup
xlog-method = fetch
retention_policy = 3 MONTHS
wal_compression = enabled
logfile = /tmp/bart.log
```

The following steps show how the required configuration parameters are determined.

Step 1: Verify that the `pg_basebackup` utility program is installed on the BART host. If you have not already done so, see Section 3.1 for instructions.

The `pg_basebackup` program in directory *`POSTGRES_INSTALL_HOME`*`/bin` is invoked by BART for taking base backups.

In the BART configuration file set the `pg_basebackup_path` parameter to the location of the `pg_basebackup` program as shown in the following example:

```
[BART]
bart-host = bartuser@192.168.2.22
backup_path = /opt/backup
pg_basebackup_path = /opt/PostgresPlus/9.4AS/bin/pg_basebackup
logfile = /tmp/bart.log
```

Step 2: Create or select the BART user account.

Determine the operating system user account that will be used to run the BART command line program. This operating system user is referred to as the *BART user account*.

If your managed database servers are mostly Advanced Servers installed in Oracle compatible mode, it would be advantageous to use `enterprisedb` as the BART user account especially if the database servers and BART are running on the same host.

If your managed database servers are mostly Advanced Servers installed in PostgreSQL compatible mode or PostgreSQL database servers, using `postgres` as the BART user account would be more convenient especially if the database servers and BART are running on the same host.

The advantage in the prior two cases is that SSH/SCP connections would not be required between BART and the local database servers.

The user account `bartuser` is used for the following examples.

Step 3: Set the environment variables for the BART user account.

When invoking BART subcommands as the BART user account chosen in Step 2, certain environment settings must be in effect:

- **LD_LIBRARY_PATH.** The `LD_LIBRARY_PATH` environment variable must include the directory containing the `libpq` library. This directory is *`POSTGRES_INSTALL_HOME`*`/lib`.
- **PATH.** Though not absolutely required, it is suggested that the `PATH` environment variable include the *`BART_HOME`*`/bin` directory. Doing so allows the BART user to invoke BART from any current working directory. Without doing

so, the BART user must invoke BART from *BART_HOME*/bin as the current working directory.

These settings can be placed in the BART user account's profile so they take effect upon login as shown by the following example:

```
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
        . ~/.bashrc
fi

# User specific environment and startup programs

export LD_LIBRARY_PATH=/opt/PostgresPlus/9.4AS/lib:$PATH
export PATH=/usr/edb-bart-1.1/bin:$PATH

PATH=$PATH:$HOME/bin

export PATH
```

Step 4: Create the BART backup catalog.

The BART user account must have access to the parent directory of the BART backup catalog as specified by the backup_path parameter in the BART configuration file.

If not, be sure the BART user account is granted the appropriate ownership and permissions to be able to create subdirectories and files within that directory.

In the following example, the BART configuration file specifies /opt/backup as the parent directory for the BART backup catalog:

```
[BART]
bart-host = bartuser@192.168.2.22
backup_path = /opt/backup
pg_basebackup_path = /opt/PostgresPlus/9.4AS/bin/pg_basebackup
logfile = /tmp/bart.log
```

The following example creates and sets the ownership and permissions on the BART backup catalog assuming bartuser is the BART user account:

```
su root
mkdir /opt/backup
chown bartuser /opt/backup
chgrp bartuser /opt/backup
chmod 700 /opt/backup
```

When you invoke the bart command line program with the INIT subcommand or in fact, with any BART subcommand, BART creates a subdirectory for each database server listed in the configuration file if the subdirectory had not already been created by BART.

Step 5: It is suggested that you leave the `logfile` configuration parameter set to its default value of `/tmp/bart.log`. This file is created the first time you invoke the `bart` command line program.

If for some reason you want to change your BART user account during the course of BART usage, you must delete the `/tmp/bart.log` file and let BART create a new log file owned by the new BART user account.

Step 6: Set any of the other optional global parameters listed at the beginning of this section. The settings of these parameters apply to all database servers, but can be overridden within each database server section.

Step 7: Add parameters for your first set of database servers in the BART configuration file. See Section 4.2.5 for the definition of the database server parameters.

4.2 Configuring a Database Server for BART Management

This section describes the procedure for enabling BART backup and recovery management for a database server. The main configuration areas are the following:

- Authorizing SSH/SCP access without a password prompt. See Section 4.2.1.
- Setting up a replication database user. See Section 4.2.2.
- Enabling WAL archiving. See Section 4.2.3.
- Updating the BART configuration file. See Section 4.2.5.

These do not have to be done in any particular order except that all must be completed before restarting the database server with WAL archiving enabled (bullet point 3).

4.2.1 Authorizing SSH/SCP Access without a Password

A fundamental mechanism of BART is the use of the Secure Shell (SSH) and the Secure Copy (SCP) Linux utility programs to copy the base backup and WAL files between the BART host and the BART managed database servers.

Note: See Section 2.1.2 for the only exception where SSH and SCP are not required.

The client/server SSH and SCP connections between the BART host and the database server hosts must not prompt for a password when establishing the connection.

A password-less connection is accomplished by the use of *authorized public keys*, which is a list of public keys of client user accounts that are to be allowed to connect to the target server.

Each client user account generates a public key, which must then be added to the target user account's authorized public keys list on the target server.

The directions are divided into the following sections:

- Section 4.2.1.1 provides an example of how you may first have to enable public key authentication usage on the server running the SSH server daemon.
- Section 4.2.1.2 provides general instructions on how to set up the authorized public keys file.
- Section 4.2.1.3 then describes the combination of hosts for BART usage on which a connection must be established without a password prompt.

Specific examples are provided in Section 6.2.

4.2.1.1 Enabling Public Key Authentication Usage

Depending upon the Linux operating system running the SSH server daemon, there may be different steps required to enable the usage of public key authentication, and hence, the capability to enable password-less SSH and SCP connections.

Check the information for your operating system for the necessary steps.

For example, for CentOS 6.5, perform the following:

In the SSH server daemon configuration file, `/etc/ssh/sshd_config`, check that the following parameter is set to `yes` and is not commented:

```
PubkeyAuthentication yes
```

Reload the configuration file:

```
[root@localhost ssh]# service sshd reload
Reloading sshd:                                            [  OK  ]
```

Any of the following commands can be used instead of `service sshd reload`:

```
service sshd stop
service sshd start
service sshd restart
```

Note: For any SSH or SCP errors or problems, examine the following log file:

```
/var/log/secure
```

4.2.1.2 Authorized Public Keys Generation

The target server to which a password-less SSH or SCP connection is to be made must contain an authorized public keys file. The file is named `authorized_keys` and is

located under the *USER_HOME*/.ssh directory where *USER_HOME* is the home directory of the user account on the target server that is to be used to establish the remote session.

On each client system that is to make a password-less connection to the target server, the client's public key is generated while logged into the client system with the user account that is to request the SSH or SCP connection. The generated public key must be then copied to the target server and concatenated onto the *USER_HOME*/.ssh/authorized_keys file.

Note: The public key should be appended onto the end of any existing authorized_keys file. Any existing authorized_keys file should not be replaced in its entirety.

The following are the general instructions for generating a client's public key file and then creating the target server's authorized public keys file.

Step 1: On the client system, log in as the user account that will be initiating the SSH or SCP connection.

Step 2: Change to the user account's home directory and check if there is an existing .ssh subdirectory. If not, create one as follows:

```
mkdir .ssh
chown user .ssh
chgrp usergroup .ssh
chmod 700 .ssh
```

Where *user* is the user account name and *usergroup* is the associated group of the user.

Step 3: Generate the public key file with the following command. Accept all prompted defaults and do not specify a passphrase when prompted for one.

```
ssh-keygen –t rsa
```

The public key file named id_rsa.pub is created in the .ssh subdirectory.

Step 4: By whatever means is available in your system environment, create a copy of file id_rsa.pub on the target server.

For example, while logged into the client where you just generated the public key file, use SCP to make a temporary copy of it on the target server:

```
scp ~/.ssh/id_rsa.pub target_user@host_address:tmp.pub
```

Step 5: Log into the target server as *target_user*, again using whatever means is possible in your system environment.

For example, while logged into the client, use SSH to log into the target server:

```
ssh target_user@host_address
```

Step 6: Change to the target user account's home directory and check if there is an existing `.ssh` subdirectory. If not, create one as shown in Step 2.

Step 7: Append the temporary, client's public key file, `tmp.pub`, to the authorized keys file named `authorized_keys`. If an existing authorized keys file does not exist, create a new file, but do not completely replace any existing authorized keys file.

```
cat tmp.pub >> ~/.ssh/authorized_keys
```

Make sure the `authorized_keys` file is only accessible by the file owner and not by groups or other users. If the `authorized_keys` file does not have the required permission setting (600) or it was newly created, change the file permissions as follows:

```
chmod 600 ~/.ssh/authorized_keys
```

Step 8: Delete the temporary public key file, `tmp.pub`.

```
rm tmp.pub
```

Now, when logged into the client system as *user* there should be no prompt for a password when commands such as the following are given:

```
ssh target_user@host_address
```

or

```
scp file_name target_user@host_address:directory_path
```

or

```
scp target_user@host_address:directory_path/file file_name
```

4.2.1.3 Required BART Connections with No Password

For BART usage, there are two scenarios requiring password-less SSH/SCP connections:

1. From each BART managed database server (SSH/SCP client) to the BART host (target SSH/SCP server) for supporting WAL archiving in the `postgresql.conf` or `postgresql.auto.conf` `archive_command` parameter.

2. From the BART host (SSH/SCP client) to each BART managed database server (target SSH/SCP server) for supporting restoration of the base backup and the archived WAL files, which occurs when the BART `RESTORE` subcommand is given.

For scenario 1, the SSH client in which the public key file (`id_rsa.pub`) is generated with the `ssh-keygen -t rsa` command is the database server. The public key file is generated by the user account running the database server.

The target SSH server in which the public key file is to be appended onto the `~/.ssh/authorized_keys` file is the BART host. The `authorized_keys` file is in the BART user account's home directory.

For scenario 2, the SSH client in which the public key file (`id_rsa.pub`) is generated with the `ssh-keygen -t rsa` command is the BART host. The public key file is generated by the BART user account.

The target SSH server in which the public key file is to be appended onto the `~/.ssh/authorized_keys` file is the database server. The `authorized_keys` file is in the home directory of the user account owning the directory where the database backup is to be restored.

See Section 6.2 for examples of each scenario.

4.2.2 Setting up a Replication Database User

For each Postgres database server that is to be managed by BART, a database user must be chosen to create base backups with the `pg_basebackup` utility program. This database user is called the *replication database user*.

The replication database user is also used by the `INIT` subcommand to set the `archive_command` configuration parameter for database server version 9.4.

The replication database user must either be a superuser or have the `replication` privilege.

If the replication database user is not a superuser, then the `CONNECT` privilege must be granted on all databases to the replication database user. This requirement may have already been satisfied by the granting of `CONNECT` privilege to `PUBLIC` on all databases. However, if this is not the case, or if the `CONNECT` privilege to `PUBLIC` has been revoked,

then the CONNECT privilege must be granted on every database to the replication database user.

If a Postgres database server you intend to manage is of version 9.4, then the replication database user for that database server must be a superuser. For these database servers, the INIT subcommand invokes the Postgres ALTER SYSTEM command to set the archive_command configuration parameter in the postgresql.auto.conf file using parameters from the BART configuration file. Use of ALTER SYSTEM requires superuser privilege. See Section 4.2.3.2 for information on setting the archive_command parameter with the INIT subcommand.

In addition, the pg_hba.conf file must include an entry permitting the replication connection for the chosen replication database user.

The following example creates a replication database non-superuser named repuser:

```
CREATE ROLE repuser WITH LOGIN REPLICATION PASSWORD 'password';
```

The following example grants the CONNECT privilege on a list of databases to the replication database non-superuser:

```
GRANT CONNECT ON DATABASE database_1, database_2 TO repuser;
```

Alternatively, the replication database user can be created as a superuser:

```
CREATE ROLE repuser WITH LOGIN SUPERUSER PASSWORD 'password';
```

The pg_hba.conf file must include a corresponding replication connection entry for repuser as shown by the last line in the following example. The replication database user must have access to database template1 as well.

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
local   all             all                                     md5
# IPv4 local connections:
host    template1       repuser         192.168.2.22/32         md5
host    all             enterprisedb    127.0.0.1/32            md5
# IPv6 local connections:
host    all             all             ::1/128                 md5
# Allow replication connections from localhost, by a user with the
# replication privilege.
host    replication     repuser         192.168.2.22/32         md5
```

The ADDRESS field for these entries contains the BART host address from where the pg_basebackup request originates.

The replication database user must be specified with the user parameter of the BART configuration file for the database server as shown by the following:

```
[PPAS93_remote]
host = 192.168.2.24
port = 5444
user = repuser
remote-host = enterprisedb@192.168.2.24
description = "PPAS 93 remote server"
```

There must be no password prompt when connecting to the database server with the replication database user. There are several Postgres standard ways to permit this. A recommended method is to use the `.pgpass` file located in the BART user account's home directory.

For example, if `bartuser` is the BART user account, then the `.pgpass` file located in `/home/bartuser/.pgpass` must contain the following entry.

```
192.168.2.24:5444:*:repuser:password
```

Thus, when `bartuser` runs BART `BACKUP`, which then invokes `pg_basebackup`, the password for the replication database user, `repuser`, is obtained from the `.pgpass` file of `bartuser` to connect to the database server running at `192.168.2.24` on port `5444`.

The `.pgpass` file must contain an entry for each BART managed database server and its corresponding replication database user and password.

4.2.3 Enabling WAL Archiving

WAL archiving must be enabled for the database server for which BART is to perform backup and recovery management.

For background information about WAL archiving see the *PostgreSQL Core Documentation* available at:

http://www.postgresql.org/docs/9.5/static/continuous-archiving.html

Section 4.2.3.1 describes the general configuration process, which is applicable to any database server version, and in fact, must be used for database server version 9.3 and earlier versions.

For database server version 9.4, see Section 4.2.3.2 for an alternative approach.

4.2.3.1 WAL Archiving Configuration

The following configuration parameters must be set in the `postgresql.conf` file to enable WAL archiving:

- Set `wal_level` to `archive`.
- Set `archive_mode` to `on`.

- Set the `archive_command` to copy the WAL files to the BART backup catalog.
- Set `max_wal_senders` to a value high enough to leave at least one session available for the backup. If the `xlog-method=stream` parameter setting is to be used by this database server as determined in the BART configuration file, the `max_wal_senders` setting must account for an additional session for the transaction log streaming (that is, the setting must be a minimum of `2`). See sections 4.1 and 4.2.5 for information on the `xlog-method` parameter.

Note: The `archive_command` configuration parameter discussed in this section is located in the `postgresql.conf` file. This *Postgres* `archive_command` parameter is used in a different manner than the *BART* `archive_command` parameter, which may be set in the server sections of the BART configuration file, but only under certain conditions. Do not include the BART `archive_command` parameter in the server section for any database server of version 9.3 or earlier as this has no effect on the WAL archiving configuration. See Section 4.2.3.2 for instructions on how the BART `archive_command` parameter can be used for database version 9.4.

The `ARCHIVE PATH` field displayed by the BART `SHOW-SERVERS` subcommand shows the full directory path where the WAL files should be copied as specified in the Postgres `archive_command` configuration parameter in the `postgresql.conf` file:

```
-bash-4.1$ bart SHOW-SERVERS -s ppas93_remote
SERVER NAME          : ppas93_remote
HOST NAME            : 192.168.2.24
USER NAME            : repuser
PORT                 : 5444
REMOTE HOST          :
RETENTION POLICY     : none
DISK UTILIZATION     : 0.00 bytes
NUMBER OF ARCHIVES   : 0
ARCHIVE PATH         : /opt/backup/ppas93_remote/archived_wals
ARCHIVE COMMAND      : (disabled)
XLOG METHOD          : fetch
WAL COMPRESSION      : disabled
TABLESPACE PATH(s)   :
DESCRIPTION          : "PPAS 93 remote server"
```

In the following example, SCP copies the WAL files to directory `/opt/backup/ppas93_remote/archived_wals` at the BART host located at `192.168.2.22` as the `bartuser` user account.

Using the `bartuser` account allows the operation to copy to the BART backup catalog owned by `bartuser`.

```
wal_level = archive                     # minimal, archive, or hot_standby
                                        # (change requires restart)
    .
    .
    .
archive_mode = on               # allows archiving to be done
                                # (change requires restart)
```

```
archive_command = 'scp %p
bartuser@192.168.2.22:/opt/backup/ppas93 remote/archived wals/%f'
                                # command to use to archive a logfile segment
                                # placeholders: %p = path of file to archive
                                #               %f = file name only

    .
    .
    .
max wal senders = 1             # max number of walsender processes
                                # (change requires restart)
```

The database server must be restarted in order to initiate WAL archiving, but do not do so until you have verified that the full path of the BART backup catalog has been created by some prior BART subcommand, otherwise the archive operation will fail.

4.2.3.2 Archive Command Auto Configuration

Note: The instructions in this section can only be used for Postgres database server version 9.4 and beyond.

For database server version 9.4, the Postgres `archive_command` parameter can be automatically configured with the `INIT` subcommand. The `INIT` subcommand invokes the Postgres `ALTER SYSTEM` command to set the Postgres `archive_command` configuration parameter in the `postgresql.auto.conf` file located in the managed database server's *`POSTGRES_INSTALL_HOME`*`/data` directory. See Section 5.4.1 for additional information on the `INIT` subcommand.

The archive command string that the `INIT` subcommand generates into the `postgresql.auto.conf` file is determined by the parameter setting of the BART `archive_command` parameter located in the BART configuration file.

The server section of the BART configuration file can contain a BART `archive_command` parameter to specify the desired format of the archive command string to be generated into the Postgres `archive_command` parameter in the `postgresql.auto.conf` file. If the BART `archive_command` parameter is not set in the server section for a given database server, the command string that is configured uses the following default format:

```
scp %p %h:%a/%f
```

where:

`%p`

Path of the file to archive used by the Postgres archiving process

`%h`

Replaced by the setting of the `bart-host` parameter located in the global section of the BART configuration file

`%a`

Replaced by the archive path of where the WAL files are to be stored. The archive path takes the form *`backup_path`*`/`*`server_name`*`/archived_wals` where *`backup_path`* is the BART backup catalog parent directory specified in the global section of the BART configuration file and *`server_name`* is the lowercase conversion of the database server name specified for this database server in the server section of the BART configuration file.

`%f`

Archived file name used by the Postgres archiving process

The placeholders `%h` and `%a` are replaced by the `INIT` subcommand when creating the archive command string. The placeholders `%p` and `%f` are not replaced by the `INIT` subcommand, but are kept as given to be used by the Postgres archiving process.

For example, to use the default archive command format, the BART configuration file contains the following settings where the BART `archive_command` parameter is omitted from the server section for `PPAS94`:

```
[BART]
bart-host= bartuser@192.168.2.22
backup_path = /opt/backup
pg_basebackup_path = /opt/PostgresPlus/9.4AS/bin/pg_basebackup
logfile = /tmp/bart.log

[PPAS94]
host = 127.0.0.1
port = 5444
user = repuser
description = "PPAS 94 server"
```

The `INIT` subcommand is invoked by BART user account `bartuser` as follows:

```
[bartuser@localhost ~]$ bart INIT -s ppas94 -o
INFO:  setting archive_command for server 'ppas94'
WARNING: archive_command is set. server restart is required
```

The BART backup catalog directory will be completed if it has not already been done so.

The resulting Postgres archive command string in the `postgresql.auto.conf` file located in the managed database server's *`POSTGRES_INSTALL_HOME`*`/data` directory appears as follows:

```
# Do not edit this file manually!
# It will be overwritten by ALTER SYSTEM command.
archive_command = 'scp %p bartuser@192.168.2.22:/opt/backup/ppas94/archived_wals/%f'
```


Note: Run the `INIT` subcommand with the `-o` option to take advantage of the auto configuration process. This option overrides any existing Postgres `archive_command` setting in the `postgresql.conf` or the `postgresql.auto.conf` file. In addition, the `-o` option must be used to generate the command string if the `archive_mode` configuration parameter is set to `off` even if there are no existing settings of the Postgres `archive_command` in the `postgresql.conf` or `postgresql.auto.conf` files.

Note: If you set the Postgres `archive_command` parameter in the `postgresql.conf` file as described in Section 4.2.3.1 and wish to use that particular setting, do not specify the `-o` option when running the `INIT` subcommand, otherwise the Postgres `archive_command` setting is overridden according to the auto configuration process described in this section.

You may use an archive command other than the default by setting the BART `archive_command` parameter with the desired command string in the server section of the BART configuration file.

In this example, the following BART configuration file is used with an explicit setting of the BART `archive_command` parameter:

```
[BART]
bart-host= enterprisedb@192.168.2.22
backup_path = /opt/backup
pg_basebackup_path = /opt/PostgresPlus/9.4AS/bin/pg_basebackup
logfile = /tmp/bart.log

[PPAS94]
host = 127.0.0.1
port = 5444
user = repuser
archive_command = 'cp %p %a/%f'
description = "PPAS 94 server"
```

The `INIT` subcommand is invoked by BART user account `enterprisedb` as follows:

```
-bash-4.1$ bart INIT -s ppas94 -o
INFO:  setting archive_command for server 'ppas94'
WARNING: archive_command is set. server restart is required
```

The resulting Postgres `archive_command` parameter in the `postgresql.auto.conf` file appears as follows:

```
# Do not edit this file manually!
# It will be overwritten by ALTER SYSTEM command.
archive_command = 'cp %p /opt/backup/ppas94/archived_wals/%f'
```

After generating the desired command string in the `postgresql.auto.conf` file, complete the required WAL archive settings in the `postgresql.conf` file:

- Set `wal_level` to `archive`.

- Set `archive_mode` to `on`.
- Set `max_wal_senders` to a value high enough to leave at least one session available for the backup. If the `xlog-method=stream` parameter setting is to be used by this database server as determined in the BART configuration file, the `max_wal_senders` setting must account for an additional session for the transaction log streaming (that is, the setting must be a minimum of `2`). See sections 4.1 and 4.2.5 for information on the `xlog-method` parameter.

Restart the database server when you are ready to initiate WAL archiving.

When the database server has been restarted, the `ARCHIVE COMMAND` field of the `SHOW-SERVERS` subcommand displays the active Postgres archive command as shown by the following:

```
-bash-4.1$ bart SHOW-SERVERS -s ppas94
SERVER NAME         : ppas94
HOST NAME           : 127.0.0.1
USER NAME           : repuser
PORT                : 5444
REMOTE HOST         :
RETENTION POLICY    : none
DISK UTILIZATION    : 48.00 MB
NUMBER OF ARCHIVES  : 0
ARCHIVE PATH        : /opt/backup/ppas94/archived_wals
ARCHIVE COMMAND     : cp %p /opt/backup/ppas94/archived_wals/%f
XLOG METHOD         : fetch
WAL COMPRESSION     : disabled
TABLESPACE PATH(s)  :
DESCRIPTION         : "PPAS 94 server"
```

4.2.4 Using Tablespaces

Note: If the database cluster contains user-defined tablespaces (that is, tablespaces created with the `CREATE TABLESPACE` command), only the tar backup file format is supported. In other words, the `BACKUP` subcommand option `-F p` for specifying a plain text backup file format may not be used. Consequently, transaction log streaming with the `BACKUP` subcommand cannot be used. Thus, the `xlog-method = fetch` parameter setting must be in effect for such database clusters containing user-defined tablespaces.

If the particular database cluster you plan to back up contains tablespaces created by the `CREATE TABLESPACE` command, be sure to set the `tablespace_path` parameter in the BART configuration file before you perform a BART `RESTORE` operation.

The `tablespace_path` parameter specifies the directory paths to which you want the tablespaces to be restored.

The `tablespace_path` parameter takes the following format:

```
OID_1=tablespace_path_1;OID_2=tablespace_path_2 ...
```

OID_1, *OID_2*, … are the Object Identifiers of the tablespaces. You can find the OIDs of the tablespaces and their corresponding soft links to the directories by listing the contents of the *POSTGRES_INSTALL_HOME*/data/pg_tblspc subdirectory as shown in the following example:

```
[root@localhost pg_tblspc]# pwd
/opt/PostgresPlus/9.3AS/data/pg_tblspc
[root@localhost pg_tblspc]# ls -l
total 0
lrwxrwxrwx 1 enterprisedb enterprisedb 17 Aug 22 16:38 16644 -> /mnt/tablespace_1
lrwxrwxrwx 1 enterprisedb enterprisedb 17 Aug 22 16:38 16645 -> /mnt/tablespace_2
```

The OIDs are 16644 and 16645 to directories /mnt/tablespace_1 and /mnt/tablespace_2, respectively. If you later wish to restore the tablespaces to the same locations as indicated in the preceding example, the BART configuration file must contain the following entry:

```
[PPAS93]
host = 127.0.0.1
port = 5444
user = enterprisedb
tablespace_path = 16644=/mnt/tablespace_1;16645=/mnt/tablespace_2
description = "PPAS 93 Server"
```

If you wish to restore the tablespaces to different locations, specify the new directory locations in the tablespace_path parameter.

In either case, the directories specified in the tablespace_path parameter must exist and be empty at the time you perform the BART RESTORE operation.

If the database server is running on a remote host (in other words you are also using the remote-host configuration parameter or will specify the --remote-host option with the RESTORE subcommand), the specified tablespace directories must exist on the specified remote host.

The directories must be owned by the user account with which you intend to start the database server (typically the Postgres user account) with no access by other users or groups as is required for the directory path to which the main base backup is to be restored.

Example

The following is an example of backing up and then restoring a database cluster on a remote host that includes tablespaces.

Note: This example emphasizes the steps that are affected by tablespace usage. See Chapter 5 for the complete process required for backing up and restoring a database cluster.

On an Advanced Server database running on a remote host, the following tablespaces are created and used by two tables:

```
edb=# CREATE TABLESPACE tblspc_1 LOCATION '/mnt/tablespace_1';
CREATE TABLESPACE
edb=# CREATE TABLESPACE tblspc_2 LOCATION '/mnt/tablespace_2';
CREATE TABLESPACE
edb=# \db
              List of tablespaces
    Name    |    Owner     |     Location
------------+--------------+------------------
 pg_default | enterprisedb |
 pg_global  | enterprisedb |
 tblspc_1   | enterprisedb | /mnt/tablespace_1
 tblspc_2   | enterprisedb | /mnt/tablespace_2
(4 rows)

edb=# CREATE TABLE tbl_tblspc_1 (c1 TEXT) TABLESPACE tblspc_1;
CREATE TABLE
edb=# CREATE TABLE tbl_tblspc_2 (c1 TEXT) TABLESPACE tblspc_2;
CREATE TABLE
edb=# \d tbl_tblspc_1
Table "enterprisedb.tbl_tblspc_1"
 Column | Type | Modifiers
--------+------+-----------
 c1     | text |
Tablespace: "tblspc_1"

edb=# \d tbl_tblspc_2
Table "enterprisedb.tbl_tblspc_2"
 Column | Type | Modifiers
--------+------+-----------
 c1     | text |
Tablespace: "tblspc_2"
```

The following shows the OIDs assigned to the tablespaces and the symbolic links to the tablespace directories:

```
-bash-4.1$ pwd
/opt/PostgresPlus/9.3AS/data/pg_tblspc
-bash-4.1$ ls -l
total 0
lrwxrwxrwx 1 enterprisedb enterprisedb 17 Nov 16 16:17 16587 -> /mnt/tablespace_1
lrwxrwxrwx 1 enterprisedb enterprisedb 17 Nov 16 16:17 16588 -> /mnt/tablespace_2
```

The BART configuration file contains the following settings. Note that the `tablespace_path` parameter does not have to be set at this point.

```
[BART]
bart-host= enterprisedb@192.168.2.22
backup_path = /opt/backup
pg_basebackup_path = /opt/PostgresPlus/9.4AS/bin/pg_basebackup
logfile = /tmp/bart.log

[PPAS93_remote]
host = 192.168.2.24
port = 5444
user = repuser
remote-host = enterprisedb@192.168.2.24
tablespace_path =
```

```
description = "PPAS 93 remote server"
```

After the necessary configuration steps described in this chapter are performed so BART can manage the remote database server, a base backup is taken. See Chapter 5 for the preparation steps and action for taking the base backup.

```
-bash-4.1$ bart BACKUP -s ppas93_remote

INFO:  creating backup for server 'ppas93_remote'
INFO:  backup identifier: '1447709811516'
54521/54521 kB (100%), 3/3 tablespaces

INFO:  backup completed successfully
INFO:  backup checksum: 594f69fe7d26af991d4173d3823e174f of 16587.tar
INFO:  backup checksum: 7a5507567729a21c98a15c948ff6c015 of base.tar
INFO:  backup checksum: ae8c62604c409635c9d9e82b29cc0399 of 16588.tar
INFO:
BACKUP DETAILS:
BACKUP STATUS: active
BACKUP IDENTIFIER: 1447709811516
BACKUP NAME: none
BACKUP LOCATION: /opt/backup/ppas93_remote/1447709811516
BACKUP SIZE: 53.25 MB
BACKUP FORMAT: tar
XLOG METHOD: fetch
BACKUP CHECKSUM(s): 3
 ChkSum                           File
 594f69fe7d26af991d4173d3823e174f   16587.tar
 7a5507567729a21c98a15c948ff6c015   base.tar
 ae8c62604c409635c9d9e82b29cc0399   16588.tar

TABLESPACE(s): 2
 Oid     Name       Location
 16587   tblspc_1   /mnt/tablespace_1
 16588   tblspc_2   /mnt/tablespace_2

START WAL LOCATION: 000000010000000000000000F
BACKUP METHOD: streamed
BACKUP FROM: master
START TIME: 2015-11-16 16:36:51 EST
STOP TIME: 2015-11-16 16:36:52 EST
TOTAL DURATION: 1 sec(s)
```

Note in the output from the preceding example that checksums are generated for the tablespaces as well as the base backup.

Within the base backup subdirectory `1447709811516` of the BART backup catalog, the tablespace data is stored with file names `16587.tar.gz` and `16588.tar.gz` as shown by the following:

```
-bash-4.1$ pwd
/opt/backup/ppas93_remote
-bash-4.1$ ls -l
total 8
drwx------ 2 enterprisedb enterprisedb 4096 Nov 16 16:36 1447709811516
drwx------ 2 enterprisedb enterprisedb 4096 Nov 16 16:43 archived_wals
-bash-4.1$ ls -l 1447709811516
total 54536
```

```
-rw-rw-r-- 1 enterprisedb enterprisedb    19968 Nov 16 16:36 16587.tar
-rw-rw-r-- 1 enterprisedb enterprisedb    19968 Nov 16 16:36 16588.tar
-rw-rw-r-- 1 enterprisedb enterprisedb      949 Nov 16 17:05 backupinfo
-rw-rw-r-- 1 enterprisedb enterprisedb 55792640 Nov 16 16:36 base.tar
```

When you are ready to restore the backup, in addition to creating the directory to which the main database cluster is to be restored, prepare the directories to which the tablespaces are to be restored.

On the remote host, directories `/opt/restore_tblspc_1` and `/opt/restore_tblspc_2` are created and assigned the proper ownership and permissions as shown by the following. The main database cluster is to be restored to `/opt/restore`.

```
[root@localhost opt]# mkdir restore_tblspc_1
[root@localhost opt]# chown enterprisedb restore_tblspc_1
[root@localhost opt]# chgrp enterprisedb restore_tblspc_1
[root@localhost opt]# chmod 700 restore_tblspc_1
[root@localhost opt]# mkdir restore_tblspc_2
[root@localhost opt]# chown enterprisedb restore_tblspc_2
[root@localhost opt]# chgrp enterprisedb restore_tblspc_2
[root@localhost opt]# chmod 700 restore_tblspc_2
[root@localhost opt]# ls -l
total 20
drwxr-xr-x  3 root         daemon       4096 Nov 10 15:38 PostgresPlus
drwx------  2 enterprisedb enterprisedb 4096 Nov 16 17:40 restore
drwx------  2 enterprisedb enterprisedb 4096 Nov 16 17:40 restore_tblspc_1
drwx------  2 enterprisedb enterprisedb 4096 Nov 16 17:41 restore_tblspc_2
drwxr-xr-x. 2 root         root         4096 Nov 22  2013 rh
```

Set the `tablespace_path` parameter in the BART configuration file to specify the tablespace directories.

Also note that the remote host user and IP address are specified by the `remote-host` configuration parameter.

```
[PPAS93_remote]
host = 192.168.2.24
port = 5444
user = repuser
remote-host = enterprisedb@192.168.2.24
tablespace_path = 16587=/opt/restore_tblspc_1;16588=/opt/restore_tblspc_2
description = "PPAS 93 remote server"
```

The following shows invocation of the `RESTORE` subcommand:

```
-bash-4.1$ bart RESTORE -s ppas93_remote -i 1447709811516 -p /opt/restore
INFO:  restoring backup '1447709811516' of server 'ppas93_remote'
INFO:  restoring backup to enterprisedb@192.168.2.24:/opt/restore
INFO:  base backup restored
INFO:  archiving is disabled
INFO:  tablespace(s) restored
```

The following shows the restored base backup including the restored tablespaces:

```
bash-4.1$ pwd
/opt
-bash-4.1$ ls -l restore
total 104
-rw------- 1 enterprisedb enterprisedb   206 Nov 16 16:36 backup_label.old
drwx------ 6 enterprisedb enterprisedb  4096 Nov 10 15:38 base
drwx------ 2 enterprisedb enterprisedb  4096 Nov 16 17:46 global
drwx------ 2 enterprisedb enterprisedb  4096 Nov 10 15:38 pg_clog
-rw------- 1 enterprisedb enterprisedb  4438 Nov 10 16:23 pg_hba.conf
-rw------- 1 enterprisedb enterprisedb  1636 Nov 10 15:38 pg_ident.conf
drwxr-xr-x 2 enterprisedb enterprisedb  4096 Nov 16 17:45 pg_log
drwx------ 4 enterprisedb enterprisedb  4096 Nov 10 15:38 pg_multixact
drwx------ 2 enterprisedb enterprisedb  4096 Nov 16 17:45 pg_notify
drwx------ 2 enterprisedb enterprisedb  4096 Nov 10 15:38 pg_serial
drwx------ 2 enterprisedb enterprisedb  4096 Nov 10 15:38 pg_snapshots
drwx------ 2 enterprisedb enterprisedb  4096 Nov 16 17:47 pg_stat
drwx------ 2 enterprisedb enterprisedb  4096 Nov 16 17:47 pg_stat_tmp
drwx------ 2 enterprisedb enterprisedb  4096 Nov 10 15:38 pg_subtrans
drwx------ 2 enterprisedb enterprisedb  4096 Nov 16 17:42 pg_tblspc
drwx------ 2 enterprisedb enterprisedb  4096 Nov 10 15:38 pg_twophase
-rw------- 1 enterprisedb enterprisedb     4 Nov 10 15:38 PG_VERSION
drwx------ 3 enterprisedb enterprisedb  4096 Nov 16 17:47 pg_xlog
-rw------- 1 enterprisedb enterprisedb 23906 Nov 16 17:42 postgresql.conf
-rw------- 1 enterprisedb enterprisedb    61 Nov 16 17:45 postmaster.opts
-bash-4.1$
-bash-4.1$ ls -l restore_tblspc_1
total 4
drwx------ 3 enterprisedb enterprisedb 4096 Nov 16 16:18 PG_9.3_201306121
-bash-4.1$ ls -l restore_tblspc_2
total 4
drwx------ 3 enterprisedb enterprisedb 4096 Nov 16 16:18 PG_9.3_201306121
```

The symbolic links in the `pg_tblspc` subdirectory point to the restored directory location:

```
bash-4.1$ pwd
/opt/restore/pg_tblspc
-bash-4.1$ ls -l
total 0
lrwxrwxrwx 1 enterprisedb enterprisedb 21 Nov 16 17:42 16587 -> /opt/restore_tblspc_1
lrwxrwxrwx 1 enterprisedb enterprisedb 21 Nov 16 17:42 16588 -> /opt/restore_tblspc_2
```

Queries within `psql` also show the restored tablespaces:

```
edb=# \db
              List of tablespaces
    Name    |    Owner     |       Location
------------+--------------+-----------------------
 pg_default | enterprisedb |
 pg_global  | enterprisedb |
 tblspc_1   | enterprisedb | /opt/restore_tblspc_1
 tblspc_2   | enterprisedb | /opt/restore_tblspc_2
(4 rows)
```

4.2.5 Adding a Database Server to the BART Configuration File

In order for BART to manage the backup and recovery of a database server, there must be an entry for it in the server section of the BART configuration file.

The following parameters apply to the database servers in the server section:

- **[ServerName].** Identifies an entry for a database server in the server section of the configuration file. This is the name by which you refer to the database server using BART. The name is case-insensitive when referenced with BART subcommand options. A lowercase conversion of this name is used to create a subdirectory in the BART backup catalog for storing the base backups and WAL files for this database server. This parameter is required.
- **backup-name.** Template for user-defined, friendly names to be assigned to the base backups of the database server. The template is an alphanumeric string that may include the following variables to be substituted by the timestamp values when the backup is taken: 1) `%year` – 4-digit year, 2) `%month` – 2-digit month, 3) `%day` – 2-digit day, 4) `%hour` – 2-digit hour, 5) `%minute` – 2-digit minute, and 6) `%second` – 2-digit second. To include the percent sign (`%`) as a character in the backup name, specify `%%` in the template. Do not enclose the template string in quotes even if you want the template to include space characters, otherwise the enclosing quotes are stored as part of the backup name. Use of space characters in the backup name, however, then requires enclosing the backup name in quotes when referenced with the `-i` option by BART subcommands. This parameter can be overridden by the `--backup-name` option of the `BACKUP` subcommand. If this parameter is omitted from the BART configuration file, and the `--backup-name` option with a user-defined name is not specified with the `BACKUP` subcommand, then the base backup can only be referenced in BART subcommands by the BART assigned, integer backup identifier.
- **host.** IP address of the database server to be configured for backup. This parameter is required.
- **port.** Port number identifying the database server instance (that is, the relevant database cluster) to be backed up. This parameter is optional. If omitted, the default is port 5444.
- **user.** Replication database user name used by BART to establish the connection to the database server when running `pg_basebackup` or when running the `INIT` subcommand to set the Postgres `archive_command` configuration parameter for database server version 9.4. This database user must either be a superuser or have the `replication` privilege. **Note:** While running as the BART user account, the connection to the database server using this database user must not prompt for a password. Also, the `pg_hba.conf` file must contain a replication connection entry for this database user name. See Section 4.2.2 for more information. This parameter is required.
- **archive_command.** This is the BART `archive_command` parameter, which only has an effect on database server version 9.4. The content and variables

specified in the BART `archive_command` result in the archive command string to be generated into the Postgres `archive_command` configuration parameter when the `INIT` subcommand is used. Enclose the command string within single quotes (`'`). If omitted, the default is `'scp %p %h:%a/%f'`. Variables are the following: 1) `%p` – path of the file to archive used by the Postgres archiving process, 2) `%h` – replaced by the `bart-host` parameter setting, 3) `%a` – replaced by the BART archive path, and 4) `%f` – archived file name used by the Postgres archiving process. See Section 4.2.3.2 for additional information. **Note:** Do not include this parameter for database servers prior to version 9.4 as setting this parameter has no effect on the final setting of the Postgres `archive_command` parameter for database servers prior to version 9.4.

- **remote-host.** IP address of the remote server to which a backup is to be restored. The value for this parameter must be specified in the form *`remote_user`*`@`*`remote_host_address`* where *`remote_user`* is the user account on the target database server host that accepts a password-less SSH/SCP login connection and is the owner of the directory where the backup is to be restored. *`remote_host_address`* is the IP address of the remote host. For restoring a backup to a remote host, either this parameter must be set, or it may be specified as an option with the BART `RESTORE` subcommand.
- **tablespace_path.** Paths to which tablespaces are to be restored specified in format *`OID=tablespace_path;OID=tablespace_path`* … If the backup is to be restored to a remote host such as specified by the `remote-host` parameter, then the tablespace paths must exist on the remote host. This parameter is optional.
- **retention_policy.** Determines when an active backup should be marked as obsolete, and hence, be a candidate for deletion. The setting can be either *`max_number`* `BACKUPS`, *`max_number`* `DAYS`, *`max_number`* `WEEKS`, or *`max_number`* `MONTHS` where *`max_number`* is a positive integer. If all of the keywords `BACKUPS`, `DAYS`, `WEEKS`, and `MONTHS` are omitted, the specified integer is interpreted as *`max_number`* `BACKUPS` by default. The setting of the `retention_policy` parameter in the server section of the BART configuration file overrides the setting of `retention_policy` in the global section. If omitted in the server section, the setting of `retention_policy` in the global section is used. If the `retention_policy` parameter is not specified in either section, then no additional backups are marked as obsolete when the `MANAGE` subcommand is used. See Section 5.2 for information on managing backups using a retention policy.
- **xlog-method.** Determines how the transaction log is collected during execution of `pg_basebackup` through the `BACKUP` subcommand. Set to `fetch` to collect the transaction log files after the backup has completed. Set to `stream` to stream the transaction log in parallel with the base backup creation. If `stream` is used, the `max_wal_senders` configuration parameter in the `postgresql.conf` file for this database server must account for an additional session for the streaming of the transaction log, (that is, the setting must be a minimum of `2`). The setting of the `xlog-method` parameter in the server section of the BART configuration file

overrides the setting of `xlog-method` in the global section. If omitted in the server section, the setting of `xlog-method` in the global section is used. If the `xlog-method` parameter is not specified in either section, the default is `fetch`. **Note:** The `xlog-method = stream` setting must not be applied to any database server containing user-defined tablespaces. See Section 4.2.4 for additional information.

- **wal_compression.** Enables the compression of archived WAL files in the BART backup catalog when the `MANAGE` subcommand is invoked. Set to `enabled` to compress the archived WALS files in gzip format. Set to `disabled` to leave the files uncompressed. **Note:** The gzip compression program must be in the BART user account's `PATH`. The setting of the `wal_compression` parameter in the server section of the BART configuration file overrides the setting of `wal_compression` in the global section. If omitted in the server section, the setting of `wal_compression` in the global section is used. If the `wal_compression` parameter is not specified in either section, the default is `disabled`. See Section 5.4.6 for information on using the `MANAGE` subcommand for WAL compression.
- **copy_wals_during_restore.** Determines how the archived WAL files are collected when invoking the `RESTORE` subcommand. Set to `enabled` to copy the archived WAL files from the BART backup catalog to the *`restore_path`*`/archived_wals` directory prior to the database server archive recovery. Set to `disabled` to retrieve the archived WAL files directly from the BART backup catalog during the database server archive recovery. The BART generated `restore_command` parameter in the `recovery.conf` file reflects which of the two options is used. If the `RESTORE` subcommand is invoked with the `-c` option, then the archived WAL files are copied from the BART backup catalog to the *`restore_path`*`/archived_wals` directory, thus overriding any setting of the `copy_wals_during_restore` parameter in the BART configuration file. If the `RESTORE` subcommand is invoked without the `-c` option, then the following determines how the archived WAL files are restored: The setting of the `copy_wals_during_restore` parameter in the server section of the BART configuration file overrides the setting of `copy_wals_during_restore` in the global section. If omitted in the server section, the setting of `copy_wals_during_restore` in the global section is used. If the `copy_wals_during_restore` parameter is not explicitly set in either section, the default is `disabled`. See Section 5.4.7 for additional information.
- **description.** Description of the database server. This parameter is optional.

Edit the BART configuration file and add an entry for each database server.

The following is an example of three database servers:

```
[PPAS94]
host = 127.0.0.1
port = 5444
```



```
user = enterprisedb
backup-name = ppas94_%year-%month-%dayT%hour:%minute:%second
archive_command = 'cp %p %a/%f'
retention_policy = 8 BACKUPS
xlog-method = stream
description = "PPAS 94 server"

[PPAS93_remote]
host = 192.168.2.24
port = 5444
user = repuser
remote-host = enterprisedb@192.168.2.24
description = "PPAS 93 remote server"

[PG94]
host = 127.0.0.1
port = 5432
user = postgres
retention_policy = 4 DAYS
description = "PostgreSQL 94 server"
```

The following is an example of a complete BART configuration file:

```
#[BART]            # BART specifies the global section. settings in this section are
                   # valid for all servers. (Note that global section must be named
                   # BART).
#bart-host =       # Required. (value in "USER@HOST" format required). IP address of
                   # backup server where utility resides.
#backup_path =     # Required. defines the path where all of the backups will be
                   # stored.
#pg_basebackup_path = # Required. defines the pg_basebackup path.
#xlog-method =     # Optional. Valid options are [fetch | stream]. defaults to fetch.
#retention_policy= # Optional. applied to all servers.
                   # valid options are [BACKUPS | DAYS | WEEKS | MONTHS]
#logfile =         # Optional. Path to log file.
#wal compression= [enabled | disabled]
                   # Optional. defaults to disabled. (gzip compression, gzip must
                   # be in path).
#copy_wals_during_restore = [enabled | disabled]
                   # Optional. Defaults to disabled. Do not copy WALs during RESTORE.

#[ServerName]      # server name.
#backup-name =     # Optional. Specifies friendly name for the backups. This name
                   # can contain %variables that will be replaced with appropriate
                   # values at the time of backup. Following %variables are
                   # supported. (%% will be replaced with %).
                   # %year - 4 digit year
                   # %month - 2 digit month
                   # %day - 2 digit day of month
                   # %hour - 2 digit hour
                   # %minute - 2 digit minutes
                   # %second - 2 digit seconds.

#host =            # Required. IP address of the server being configured for backup.
#user =            # Required. database user name.
#port =            # Optional. default 5444.
#archive_command = # Optional. i.e 'scp %p %h:%a/%f' command to be used for
                   # archiving. This supports two format specifiers %h and %a
                   # %h will be replaced with bart-host by the utility.
                   # %a will be replaced with archive path by the utility.
                   # other identifiers are left alone to be utilized by server
                   # such as %f and %p

#remote-host =     # Optional. (value in "USER@HOST" format required). IP address
                   # of the server on which given backup will be restored.
#tablespace_path =  [OID=PATH];[OID=PATH]...
                   # Optional. path to directory where table spaces will be
```

```
                    # restored. (if the remote-host is provided then this path
                    # should be valid on that server.)
                    # required format is: tablespace_path=OID=PATH;OID=PATH;...
#retention policy= # Optional. applied to specific server.
                    # valid options are [BACKUPS | DAYS | WEEKS | MONTHS]
#xlog-method =      #Optional. Valid options are [fetch | stream]. defaults to global
                    #set policy.
#wal compression= [enabled | disabled]
                    # Optional. defaults to disabled. (gzip compression, gzip must
                    # be in path).
#copy_wals_during_restore = [enabled | disabled]
                    # Optional. Defaults to disabled. Decide either to copy or stream
                    # WALs during RESTORE.
#description =      # Optional. description of the server.

[BART]
bart-host= enterprisedb@192.168.2.22
backup_path = /opt/backup
pg_basebackup_path = /opt/PostgresPlus/9.4AS/bin/pg_basebackup
logfile = /tmp/bart.log

[PPAS94]
host = 127.0.0.1
port = 5444
user = enterprisedb
backup-name = ppas94_%year-%month-%dayT%hour:%minute:%second
archive command = 'cp %p %a/%f'
retention_policy = 8 BACKUPS
xlog-method = stream
description = "PPAS 94 server"

[PPAS93_remote]
host = 192.168.2.24
port = 5444
user = repuser
remote-host = enterprisedb@192.168.2.24
description = "PPAS 93 remote server"

[PG94]
host = 127.0.0.1
port = 5432
user = postgres
retention_policy = 4 DAYS
description = "PostgreSQL 94 server"
```

Once you have the BART host and the database servers configured, you can start using BART as described in Chapter 5.

5 Operation

This chapter describes how to perform backup and recovery management operations using BART.

- Section 5.1 presents an overview of the BART management process.
- Section 5.2 describes managing backups using a retention policy.
- Section 5.3 describes the basic usage of the BART command line program and its subcommands.
- Section 5.4 provides a description of each BART subcommand.

Review these sections before proceeding with any BART operation.

5.1 BART Management Overview

After performing the configuration process described in Chapter 4, you can begin the BART backup and recovery management process.

First, perform the following steps:

1. If you have not already done so, run the `INIT` subcommand to finish creation of the BART backup catalog, which results in the complete directory structure to which base backups and WAL files are saved. This step must be done before restarting the database servers with enabled WAL archiving, otherwise the copy operation in the `archive_command` parameter of the `postgresql.conf` file or the `postgresql.auto.conf` file fails due to the absence of the target archive directory. When the directory structure is complete, the lowest level subdirectory named *`server_name`*`/archived_wals` should exist for each database server.

2. Start the Postgres database servers with archiving enabled. Verify that the WAL files are appearing in the subdirectories named *`server_name`*`/archived_wals` in the BART backup catalog for each database server. (The archiving frequency is dependent upon other `postgresql.conf` configuration parameters.) Check the Postgres database server log files to ensure there are no archiving errors. Archiving should be operational before taking a base backup in order to ensure that the WAL files that may be created during the base backup process are archived.

3. Create a base backup for each database server. The base backup establishes the starting point of when point-in-time recovery can begin.

There are now a number of other BART management processes you may perform:

- Verify the checksum of the base backups using the `VERIFY-CHKSUM` subcommand.

- Display database server information with the SHOW-SERVERS subcommand.
- Display base backup information with the SHOW-BACKUPS subcommand.
- Perform the BACKUP subcommand to create additional base backups.
- Compress the archived WAL files in the BART backup catalog by enabling WAL compression in the BART configuration file and then invoking the MANAGE subcommand.
- Determine and set the retention policy for backups in the BART configuration file.
- Establish the procedure for using the MANAGE subcommand to enforce the retention policy for backups. This may include using cron jobs to schedule the MANAGE subcommand.

The following are the steps to perform a point-in-time recovery operation for a database cluster:

1. Shut down the database server using the appropriate method such as `/etc/init.d/ppas-x.x stop`, *`POSTGRES_INSTALL_HOME`*`/bin/pg_ctl stop -D` *`data_directory`*, etc.

2. Decide if you want to restore the database cluster and tablespace files a) to new, empty directories; or b) to reuse the existing database cluster directories. For option (a), create the new directories with the appropriate directory ownership and permissions. For option (b), delete all the files and subdirectories in the existing directories, but it is strongly recommended that you make a copy of this data before deleting it. Be sure to save any recent WAL files in the `pg_xlog` subdirectory that have not been archived to the BART backup catalog *`server_name`*`/archived_wals` subdirectory.

3. Run the BART `SHOW-BACKUPS -s` *`server_name`* subcommand to list the base backup IDs of the database server. Use the `-t` option if you wish to display the user-defined backup names as well. You will need to supply the appropriate base backup ID or backup name with the BART RESTORE subcommand unless you intend to restore the most recent backup in which case the `-i` option of the RESTORE subcommand for specifying the backup ID or backup name may be omitted.

4. Run the BART RESTORE subcommand with the appropriate options. The base backup is restored to the directory specified by the `-p` *`restore_path`* option. In addition, if the RESTORE subcommand `-c` option is specified or if the enabled setting of the `copy_wals_during_restore` BART configuration parameter is applicable to the database server, then the required, archived WAL files from the BART backup catalog are copied to the *`restore_path`*`/archived_wals` subdirectory.

5. Copy any saved WAL files from Step 2 that were not archived to the BART backup catalog to the *restore_path*/pg_xlog subdirectory.

6. If generated for point-in-time recovery, verify that the recovery.conf file created in the directory specified with the RESTORE subcommand's -p *restore_path* option was generated with the correct recovery parameter settings. Note that if the RESTORE subcommand -c option is specified or if the enabled setting of the copy_wals_during_restore BART configuration parameter is applicable to the database server, then the restore_command parameter retrieves the archived WAL files from the *restore_path*/archived_wals subdirectory that was created by the RESTORE subcommand, otherwise the restore_command retrieves the archived WAL files from the BART backup catalog.

7. The BART RESTORE subcommand disables WAL archiving in the restored database cluster. If you want to immediately enable WAL archiving, modify the postgresql.conf file by deleting the archive_mode = off parameter that BART appends to the end of the file.

8. Start the database server, which will then perform the point-in-time recovery operation if a recovery.conf file was generated.

See Section 5.4 for a detailed description of the BART subcommands.

5.2 Managing Backups Using a Retention Policy

Over the course of time when using BART, the number of backups can grow significantly. This ultimately leads to a large consumption of disk space unless an administrator periodically performs the process of deleting the oldest backups that are no longer needed.

This process of determining when a backup is old enough to be deleted and then actually deleting such backups can be accomplished and eventually automated with the following basic steps:

1. Determine and set a retention policy in the BART configuration file. A *retention policy* is a rule that determines when a backup is considered obsolete. The retention policy can be applied globally to all servers (see Section 4.1), but each server can override the global retention policy with its own (see Section 4.2.5).

2. Use the MANAGE subcommand to categorize and manage backups according to the retention policy. Such functionality includes determining which active backups should be considered obsolete at this current point in time, selecting backups to keep indefinitely, and physically deleting obsolete backups. When an obsolete backup is deleted, its base backup taken with the BACKUP subcommand along with that backup's archived WAL files are deleted.

3. Once the retention policies have been determined and verified, you can create a cron job to periodically run the `MANAGE` subcommand to evaluate the backups and then list and/or delete the obsolete backups.

Section 5.2.1 provides an overview of the terminology and types of retention policies.

Section 5.2.2 describes the concept of marking the status of backups according to the retention policy.

Section 5.2.3 describes setting the different types of retention policies.

Section 5.2.4 describes the process of managing the backups such as marking the backup status, keeping selected backups indefinitely, listing obsolete backups, and deleting obsolete backups.

5.2.1 Overview

The BART retention policy results in the categorization of each backup in one of three statuses – *active*, *obsolete*, and *keep*:

- **Active.** The backup satisfies the retention policy applicable to its server. Such backups would be considered necessary to ensure the recovery safety for the server and thus should be retained.
- **Obsolete.** The backup does not satisfy the retention policy applicable to its server. The backup is no longer considered necessary for the recovery safety of the server and thus can be deleted.
- **Keep.** The backup is to be retained regardless of the retention policy applicable to its server. The backup is considered vital to the recovery safety for the server and thus should not be deleted for an indefinite period of time.

There are two types of retention policies:

- **Redundancy Retention Policy.** The *redundancy retention policy* relies on a specified, maximum number of most recent backups to retain for a given server. When the number of backups exceeds that maximum number, the oldest backups are considered obsolete (except for backups marked as keep). Section 5.2.3.1 describes this type of retention policy.
- **Recovery Window Retention Policy.** The *recovery window retention policy* relies on a time frame (the recovery window) for when a backup should be considered active. The boundaries defining the recovery window are the current date/time (the ending boundary of the recovery window) and the date/time going back in the past for a specified length of time (the starting boundary of the recovery window). If the date/time the backup was taken is within the recovery window (that is, the backup date/time is on or after the starting date/time of the recovery window, then the backup is considered active, otherwise it is considered

obsolete (except for backups marked as keep). Section 5.2.3.2 describes this type of retention policy.

Thus, for the recovery window retention policy, the recovery window time frame dynamically shifts so the end of the recovery window is always the current date/time when the MANAGE subcommand is run. As you run the MANAGE subcommand at future points in time, the starting boundary of the recovery window moves forward in time.

At some future point, the date/time of when a backup was taken will be earlier than the starting boundary of the recovery window. This is when an active backup's status will then be considered obsolete.

You can see the starting boundary of the recovery window at any point in time by running the SHOW-SERVERS subcommand. The RETENTION POLICY field of the SHOW-SERVERS subcommand displays the starting boundary of the recovery window.

5.2.2 Marking the Backup Status

When a backup is initially created with the BACKUP subcommand, it is always recorded with active status.

It is only by using the MANAGE subcommand at a particular point in time that active backups are evaluated to determine if their status should be changed to obsolete in accordance with the retention policy.

In addition, it is only when the MANAGE subcommand is invoked either with no options or with only the -s option (in order to specify the database server) that active backups are evaluated and also *marked* (that is, internally recorded by BART) as obsolete. See Section 5.4.6 for information on the MANAGE subcommand usage with its options.

Once a backup has been marked as obsolete, it cannot be changed back to active unless you perform the following steps:

- Using the MANAGE subcommand, specify the -c option along with the backup identifier or name with the -i option. If you wish to keep this particular backup indefinitely, use -c keep, otherwise use -c nokeep.
- If you used the -c nokeep option, the backup status is changed back to active. The next time the MANAGE subcommand is used, the backup is re-evaluated to determine if its status needs to be changed back to obsolete based upon the current retention policy in the BART configuration file.

Note that if the retention_policy parameter is set in a certain manner, you run the MANAGE subcommand to mark the backups according to that retention_policy setting, and then you change or disable the retention_policy parameter by

commenting it out, the current, marked status of the backups are probably inconsistent with the current `retention_policy` setting.

If you wish to modify the backup status to be consistent with the current `retention_policy` setting, then perform the following:

- If there are backups currently marked as obsolete that would no longer be considered obsolete under a new retention policy, run the `MANAGE` subcommand with the `-c nokeep` option for such backups to change the obsolete status to active status. You can also specify the `-i all` option, which would change all backups back to active status, including those currently marked as keep.
- Run the `MANAGE` subcommand either with no options or with only the `-s` option to reset the marked status based upon the new `retention_policy` setting in the BART configuration file.

Deletion of backups with the `MANAGE -d` option relies on the last occasion when the backups have been marked.

The current marking (the status) of the backups can be viewed with the `SHOW-BACKUPS` subcommand.

5.2.3 Setting the Retention Policy

The retention policy is determined by the `retention_policy` parameter in the BART configuration file. To set it globally for all servers, see Section 4.1. To set it by database server, see Section 4.2.5.

The two types of retention policies are the redundancy retention policy described in Section 5.2.3.1 and the recovery window retention policy described in Section 5.2.3.2.

5.2.3.1 Redundancy Retention Policy

To use the redundancy retention policy, set `retention_policy = max_number BACKUPS` where *`max_number`* is a positive integer designating the maximum number of most recent backups.

The following are some additional restrictions:

- The keyword `BACKUPS` must always be specified in plural form (for example, `1 BACKUPS`).
- BART will accept a maximum integer value of 2,147,483,647 for *`max_number`*, however, a realistic, practical value based on your system environment must always be used.

The redundancy retention policy is the default type of retention policy if all keywords BACKUPS, DAYS, WEEKS, and MONTHS following the *max_number* integer are omitted as shown by the following example:

```
retention_policy = 3
```

In the following example the redundancy retention policy setting considers the three most recent backups as the active backups. Any older backups except those marked as keep are considered obsolete.

```
[PPAS94]
host = 127.0.0.1
port = 5444
user = enterprisedb
archive_command = 'cp %p %a/%f'
retention_policy = 3 BACKUPS
description = "PPAS 94 server"
```

The SHOW-SERVERS subcommand displays the 3 backup redundancy retention policy in the RETENTION POLICY field:

```
-bash-4.1$ bart SHOW-SERVERS -s ppas94
SERVER NAME         : ppas94
HOST NAME           : 127.0.0.1
USER NAME           : enterprisedb
PORT                : 5444
REMOTE HOST         :
RETENTION POLICY    : 3 Backups
DISK UTILIZATION    : 627.04 MB
NUMBER OF ARCHIVES  : 25
ARCHIVE PATH        : /opt/backup/ppas94/archived_wals
ARCHIVE COMMAND     : cp %p /opt/backup/ppas94/archived_wals/%f
XLOG METHOD         : fetch
WAL COMPRESSION     : disabled
TABLESPACE PATH(s)  :
DESCRIPTION         : "PPAS 94 server"
```

5.2.3.2 Recovery Window Retention Policy

To use the recovery window retention policy, set the retention_policy parameter to the desired length of time for the recovery window in one of the following ways:

- Set to *max_number* DAYS to define the start date/time recovery window boundary as the number of days specified by *max_number* going back in time from the current date/time.
- Set to *max_number* WEEKS to define the start date/time recovery window boundary as the number of weeks specified by *max_number* going back in time from the current date/time.

- Set to *max_number* MONTHS to define the start date/time recovery window boundary as the number of months specified by *max_number* going back in time from the current date/time.

The following are some additional restrictions:

- The keywords DAYS, WEEKS, and MONTHS must always be specified in plural form (for example, 1 DAYS, 1 WEEKS, or 1 MONTHS).
- BART will accept a maximum integer value of 2,147,483,647 for *max_number*, however, a realistic, practical value based on your system environment must always be used.

A backup is considered active if the date/time of the backup, down to a second accuracy, is equal to or greater than the start of the recovery window date/time.

There are a couple of ways you can view the actual, calculated recovery window as described by the following.

Invoking BART in debug mode (with the -d option) using the MANAGE subcommand with the -n option displays the calculated time length of the recovery window based on the retention_policy setting and the current date/time.

For example, using the following retention_policy settings:

```
[PPAS94]
host = 127.0.0.1
port = 5444
user = enterprisedb
archive_command = 'cp %p %a/%f'
retention_policy = 3 DAYS
backup-name = ppas94_%year-%month-%dayT%hour:%minute:%second
description = "PPAS 94 server"

[PPAS94_2]
host = 127.0.0.1
port = 5445
user = enterprisedb
archive_command = 'cp %p %a/%f'
retention_policy = 3 WEEKS
description = "PPAS 94 server 2"

[PG94]
host = 127.0.0.1
port = 5432
user = postgres
retention_policy = 3 MONTHS
description = "PostgreSQL 94 server"
```

If the MANAGE subcommand is invoked in debug mode along with the -n option on 2015-04-17, the following results are displayed:

```
-bash-4.1$ bart -d MANAGE -n
```

```
DEBUG: Server: ppas94, Now: 2015-04-17 16:34:03 EDT, RetentionWindow: 259200 (secs) ==>
72 hour(s)
DEBUG: Server: ppas94_2, Now: 2015-04-17 16:34:03 EDT, RetentionWindow: 1814400 (secs)
==> 504 hour(s)
DEBUG: Server: pg94, Now: 2015-04-17 16:34:03 EDT, RetentionWindow: 7776000 (secs) ==>
2160 hour(s)
```

For server `ppas94`, 72 hours translates to a recovery window of 3 days.

For server `ppas94_2`, 504 hours translates to a recovery window of 21 days (3 weeks).

For server `pg94`, 2160 hours translates to a recovery window of 90 days (3 months).

Note: For a setting of *`max_number`* `MONTHS`, the calculated, total number of days for the recovery window is dependent upon the actual number of days in the preceding months from the current date/time. Thus, *`max_number`* `MONTHS` is not always exactly equivalent to *`max_number`* `x 30 DAYS`. (For example, if the current date/time is in the month of March, a 1-month recovery window would be equivalent to only 28 days because the preceding month is February. Thus, for a current date of March 31, a 1-month recovery window would start on March 3.) However, the typical result is that the day of the month of the starting recovery window boundary will be the same day of the month of when the `MANAGE` subcommand is invoked.

Using the `SHOW-SERVERS` subcommand, the `RETENTION POLICY` field displays the start of the recovery window.

In the following example, the recovery window retention policy setting considers the backups taken within a 3-day recovery window as the active backups.

```
[PPAS94]
host = 127.0.0.1
port = 5444
user = enterprisedb
archive_command = 'cp %p %a/%f'
retention_policy = 3 DAYS
description = "PPAS 94 server"
```

The start of the 3-day recovery window displayed in the `RETENTION POLICY` field is `2015-04-07 14:57:36 EDT` when the `SHOW-SERVERS` subcommand is invoked on `2015-04-10`.

At this current point in time, backups taken on or after `2015-04-07 14:57:36 EDT` would be considered active. Backups taken prior to `2015-04-07 14:57:36 EDT` would be considered obsolete except for backups marked as keep.

```
-bash-4.1$ date
Fri Apr 10 14:57:33 EDT 2015
-bash-4.1$
-bash-4.1$ bart SHOW-SERVERS -s ppas94
SERVER NAME         : ppas94
HOST NAME           : 127.0.0.1
```



```
USER NAME            : enterprisedb
PORT                 : 5444
REMOTE HOST          :
RETENTION POLICY     : 2015-04-07 14:57:36 EDT
DISK UTILIZATION     : 824.77 MB
NUMBER OF ARCHIVES   : 37
ARCHIVE PATH         : /opt/backup/ppas94/archived_wals
ARCHIVE COMMAND      : cp %p /opt/backup/ppas94/archived_wals/%f
XLOG METHOD          : fetch
WAL COMPRESSION      : disabled
TABLESPACE PATH(s)   :
DESCRIPTION          : "PPAS 94 server"
```

In the following example, the recovery window retention policy setting considers the backups taken within a 3-week recovery window as the active backups.

```
[PPAS94_2]
host = 127.0.0.1
port = 5445
user = enterprisedb
archive_command = 'cp %p %a/%f'
retention_policy = 3 WEEKS
description = "PPAS 94 server 2"
```

The start of the 3-week recovery window displayed in the `RETENTION POLICY` field is `2015-03-20 14:59:42 EDT` when the `SHOW-SERVERS` subcommand is invoked on `2015-04-10`.

At this current point in time, backups taken on or after `2015-03-20 14:59:42 EDT` would be considered active. Backups taken prior to `2015-03-20 14:59:42 EDT` would be considered obsolete except for backups marked as keep.

```
-bash-4.1$ date
Fri Apr 10 14:59:39 EDT 2015
-bash-4.1$
-bash-4.1$ bart SHOW-SERVERS -s ppas94_2
SERVER NAME          : ppas94_2
HOST NAME            : 127.0.0.1
USER NAME            : enterprisedb
PORT                 : 5445
REMOTE HOST          :
RETENTION POLICY     : 2015-03-20 14:59:42 EDT
DISK UTILIZATION     : 434.53 MB
NUMBER OF ARCHIVES   : 22
ARCHIVE PATH         : /opt/backup/ppas94_2/archived_wals
ARCHIVE COMMAND      : cp %p /opt/backup/ppas94_2/archived_wals/%f
XLOG METHOD          : fetch
WAL COMPRESSION      : disabled
TABLESPACE PATH(s)   :
DESCRIPTION          : "PPAS 94 server 2"
```

In the following example, the recovery window retention policy setting considers the backups taken within a 3-month recovery window as the active backups.

```
[PG94]
host = 127.0.0.1
port = 5432
```

```
user = postgres
retention_policy = 3 MONTHS
description = "PostgreSQL 94 server"
```

The start of the 3-month recovery window displayed in the `RETENTION POLICY` field is `2015-01-10 14:04:23 EST` when the `SHOW-SERVERS` subcommand is invoked on `2015-04-10`.

At this current point in time, backups taken on or after `2015-01-10 14:04:23 EST` would be considered active. Backups taken prior to `2015-01-10 14:04:23 EST` would be considered obsolete except for backups marked as keep.

```
-bash-4.1$ date
Fri Apr 10 15:04:19 EDT 2015
-bash-4.1$
-bash-4.1$ bart SHOW-SERVERS -s pg94
SERVER NAME         : pg94
HOST NAME           : 127.0.0.1
USER NAME           : postgres
PORT                : 5432
REMOTE HOST         :
RETENTION POLICY    : 2015-01-10 14:04:23 EST
DISK UTILIZATION    : 480.76 MB
NUMBER OF ARCHIVES  : 26
ARCHIVE PATH        : /opt/backup/pg94/archived_wals
ARCHIVE COMMAND     : scp %p
enterprisedb@192.168.2.22:/opt/backup/pg94/archived_wals/%f
XLOG METHOD         : fetch
WAL COMPRESSION     : disabled
TABLESPACE PATH(s)  :
DESCRIPTION         : "PostgreSQL 94 server"
```

The following section describes how to manage the backups based on the retention policy.

5.2.4 Managing the Backups

The `MANAGE` subcommand is used to evaluate and categorize backups according to the retention policy set in the BART configuration file. When a backup is first created with the `BACKUP` subcommand, it always marked as active. Upon usage of the `MANAGE` subcommand, an active backup may be marked as obsolete. Obsolete backups can then be deleted.

The various aspects of backup management are discussed in the following sections:

- Section 5.2.4.1 discusses the rules for deleting backups dependent upon the backup status and the subcommand used.
- Section 5.2.4.2 shows how to retain a backup indefinitely by marking it as keep as well as resetting backups marked as obsolete and keep back to active status.
- Section 5.2.4.3 demonstrates the general process for evaluating, marking, then deleting obsolete backups.

5.2.4.1 Deletions Permitted Under a Retention Policy

This section describes how and under what conditions backups may be deleted when under a retention policy.

It is recommended that obsolete backups be deleted using the MANAGE subcommand. The DELETE subcommand should be used only for special administrative purposes.

Important distinctions between using the MANAGE subcommand with the -d option as compared to using the DELETE subcommand are the following:

- The `MANAGE` subcommand deletion relies solely upon how a backup status is currently marked (that is, internally recorded by BART). The current setting of the `retention_policy` parameter in the BART configuration file is ignored.
- The `DELETE` subcommand relies solely upon the current setting of the `retention_policy` parameter in the BART configuration file. The current active, obsolete, or keep status of a backup is ignored.

Thus, the deletion behavior of the `MANAGE` subcommand and the `DELETE` subcommand are based on different aspects of the retention policy.

The specific deletion rules for the `MANAGE` and `DELETE` subcommands are as follows:

- The `MANAGE` subcommand with the `-d` option can only delete backups marked as obsolete. This deletion occurs regardless of the current `retention_policy` setting in the BART configuration file.
- Under a redundancy retention policy currently set with the `retention_policy` parameter in the BART configuration file, any backup regardless of its marked status, can be deleted with the `DELETE` subcommand when the backup identifier or name is specified with the `-i` option, and if the current total number of backups for the specified database server is greater than the maximum number of redundancy backups currently specified with the `retention_policy` parameter. If the total number of backups is less than or equal to the specified, maximum number of redundancy backups, then no additional backups can be deleted using `DELETE` with the `-i` *`backup`* option.
- Under a recovery window retention policy currently set with the `retention_policy` parameter in the BART configuration file, any backup regardless of its marked status, can be deleted with the `DELETE` subcommand when the backup identifier or name is specified with the `-i` option, and if the backup date/time is not within the recovery window currently specified with the `retention_policy` parameter. If the backup date/time is within the recovery window, then it cannot be deleted using `DELETE` with the `-i` *`backup`* option.

- Invoking the `DELETE` subcommand with the `-i all` option results in the deletion of all backups regardless of the retention policy and regardless of whether the status is marked as active, obsolete, or keep.

The following table summarizes the deletion rules of backups according to their marked status. An entry of "Yes" indicates the backup may be deleted under the specified circumstances. An entry of "No" indicates that the backup may not be deleted.

Table 5-1 Allowable Backup Deletion by Status

Operation	Redundancy Retention Policy			Recovery Window Retention Policy		
	Active	Obsolete	Keep	Active	Obsolete	Keep
`MANAGE -d`	No	Yes	No	No	Yes	No
`DELETE -i` *backup*	Yes (see Note 1)	Yes (see Note 1)	Yes (see Note 1)	Yes (see Note 2)	Yes (see Note 2)	Yes (see Note 2)
`DELETE -i all`	Yes	Yes	Yes	Yes	Yes	Yes

Note 1: Deletion occurs only if the total number of backups for the specified database server is greater than the specified, maximum number of redundancy backups currently set with the `redundancy_policy` parameter in the BART configuration file.

Note 2: Deletion occurs only if the backup is not within the recovery window currently set with the `redundancy_policy` parameter in the BART configuration file.

5.2.4.2 Marking Backups for Indefinite Keep Status

There may be certain backups that you wish to keep for an indefinite period of time and thus, do not wish to subject them to deletion based upon the retention policy applied to the database server.

Such backups can be marked as keep to exclude them from being marked as obsolete.

Use the `MANAGE` subcommand with the `-c keep` option to retain such backups indefinitely as shown by the following example:

```
-bash-4.1$ bart MANAGE -s ppas94 -i 1428355371389 -c keep
INFO:  changing status of backup '1428355371389' of server 'ppas94' from
'active' to 'keep'
INFO:  1 WAL file(s) changed
```

The backup status is now displayed as keep:

```
-bash-4.1$ bart SHOW-BACKUPS -s ppas94 -i 1428355371389 -t
SERVER NAME    : ppas94
BACKUP ID      : 1428355371389
BACKUP NAME    : none
BACKUP STATUS  : keep
BACKUP TIME    : 2015-04-06 17:22:53 EDT
BACKUP SIZE    : 5.71 MB
WAL(S) SIZE    : 16.00 MB
```

```
NO. OF WALS    : 1
FIRST WAL FILE : 000000010000000000000AA
CREATION TIME  : 2015-04-06 17:22:53 EDT
LAST WAL FILE  : 000000010000000000000AA
CREATION TIME  : 2015-04-06 17:22:53 EDT
```

If at some later point in time, you decide to return such a backup to a state where it can be evaluated for deletion based upon the retention policy, you must first remove its keep status by applying the `MANAGE` subcommand with the `-c nokeep` option as shown by the following example:

```
-bash-4.1$ bart MANAGE -s ppas94 -i 1428355371389 -c nokeep
INFO:  changing status of backup '1428355371389' of server 'ppas94' from
'keep' to 'active'
INFO:  1 WAL file(s) changed
```

The backup status is changed back to active:

```
-bash-4.1$ bart SHOW-BACKUPS -s ppas94 -i 1428355371389 -t
SERVER NAME    : ppas94
BACKUP ID      : 1428355371389
BACKUP NAME    : none
BACKUP STATUS  : active
BACKUP TIME    : 2015-04-06 17:22:53 EDT
BACKUP SIZE    : 5.71 MB
WAL(S) SIZE    : 16.00 MB
NO. OF WALS    : 1
FIRST WAL FILE : 000000010000000000000AA
CREATION TIME  : 2015-04-06 17:22:53 EDT
LAST WAL FILE  : 000000010000000000000AA
CREATION TIME  : 2015-04-06 17:22:53 EDT
```

The next time the `MANAGE` subcommand is invoked with either no options or with only the `-s` option, the active backup may be marked as obsolete according to the current `retention_policy` setting.

5.2.4.3 Evaluating, Marking, and Deleting Obsolete Backups

Based upon the current number of backups for the database server for a redundancy retention policy or the current date/time for a recovery window retention policy, when the `MANAGE` subcommand is invoked it evaluates active backups for the database server specified by the `-s` option or for all database servers if `-s all` is specified or the `-s` option is omitted.

Note: The status of backups currently marked as obsolete or keep are not changed. To re-evaluate such backups and then classify them, their status must first be reset back to active with the `MANAGE -c nokeep` option. See Section 5.2.2 for information.

This section illustrates the evaluation, marking, and deletion process with two examples – the first for a redundancy retention policy and the second for a recovery window retention policy.

Redundancy Retention Policy

The following example uses a redundancy retention policy as shown by the following server configuration:

```
[PPAS94]
host = 127.0.0.1
port = 5444
user = enterprisedb
archive_command = 'cp %p %a/%f'
retention_policy = 3 BACKUPS
description = "PPAS 94 server"
```

The following is the current set of backups. Note that the last backup in the list has been marked as keep.

```
-bash-4.1$ bart SHOW-BACKUPS -s ppas94
 SERVER NAME   BACKUP ID       BACKUP TIME               BACKUP SIZE   WAL(s) SIZE
WAL FILES   STATUS

 ppas94        1428768344061   2015-04-11 12:05:46 EDT   5.72 MB       48.00 MB      3
active
 ppas94        1428684537299   2015-04-10 12:49:00 EDT   5.72 MB       272.00 MB     17
active
 ppas94        1428589759899   2015-04-09 10:29:27 EDT   5.65 MB       96.00 MB      6
active
 ppas94        1428502049836   2015-04-08 10:07:30 EDT   55.25 MB      96.00 MB      6
active
 ppas94        1428422324880   2015-04-07 11:58:45 EDT   54.53 MB      32.00 MB      2
active
 ppas94        1428355371389   2015-04-06 17:22:53 EDT   5.71 MB       16.00 MB      1
keep
```

Invoke the `MANAGE` subcommand with the `-n` option to perform a dry run to observe which active backups would be changed to obsolete according to the retention policy:

```
-bash-4.1$ bart MANAGE -s ppas94 -n
INFO:  processing server 'ppas94', backup '1428768344061'
INFO:  processing server 'ppas94', backup '1428684537299'
INFO:  processing server 'ppas94', backup '1428589759899'
INFO:  processing server 'ppas94', backup '1428502049836'
INFO:  marking backup '1428502049836' as obsolete
INFO:  6 WAL file(s) marked obsolete
INFO:  processing server 'ppas94', backup '1428422324880'
INFO:  marking backup '1428422324880' as obsolete
INFO:  2 WAL file(s) marked obsolete
INFO:  processing server 'ppas94', backup '1428355371389'
```

The dry run shows that backups `1428502049836` and `1428422324880` would be marked as obsolete.

Also note that a dry run does not change the backup status. The two backups that would be considered obsolete are still marked as active:

```
-bash-4.1$ bart SHOW-BACKUPS -s ppas94
 SERVER NAME   BACKUP ID       BACKUP TIME               BACKUP SIZE   WAL(s) SIZE
WAL FILES   STATUS
```

```
 ppas94        1428768344061   2015-04-11 12:05:46 EDT   5.72 MB       48.00 MB      3
active
 ppas94        1428684537299   2015-04-10 12:49:00 EDT   5.72 MB       272.00 MB     17
active
 ppas94        1428589759899   2015-04-09 10:29:27 EDT   5.65 MB       96.00 MB      6
active
 ppas94        1428502049836   2015-04-08 10:07:30 EDT   55.25 MB      96.00 MB      6
active
 ppas94        1428422324880   2015-04-07 11:58:45 EDT   54.53 MB      32.00 MB      2
active
 ppas94        1428355371389   2015-04-06 17:22:53 EDT   5.71 MB       16.00 MB      1
keep
```

Invoke the `MANAGE` subcommand omitting the `-n` option to change and mark the status of the backups as obsolete:

```
-bash-4.1$ bart MANAGE -s ppas94
INFO:  processing server 'ppas94', backup '1428768344061'
INFO:  processing server 'ppas94', backup '1428684537299'
INFO:  processing server 'ppas94', backup '1428589759899'
INFO:  processing server 'ppas94', backup '1428502049836'
INFO:  marking backup '1428502049836' as obsolete
INFO:  6 WAL file(s) marked obsolete
INFO:  processing server 'ppas94', backup '1428422324880'
INFO:  marking backup '1428422324880' as obsolete
INFO:  2 WAL file(s) marked obsolete
INFO:  processing server 'ppas94', backup '1428355371389'
```

The obsolete backups can be observed in a number of ways. Use the `MANAGE` subcommand with the `-l` option to list the obsolete backups:

```
-bash-4.1$ bart MANAGE -s ppas94 -l
INFO:  6 WAL file(s) will be removed
SERVER NAME: ppas94
BACKUP ID: 1428502049836
BACKUP STATUS: obsolete
BACKUP TIME: 2015-04-08 10:07:30 EDT
BACKUP SIZE: 55.25 MB
WAL FILE(s): 6
 WAL FILE: 000000010000000100000003
 WAL FILE: 000000010000000100000002
 WAL FILE: 000000010000000100000001
 WAL FILE: 000000010000000100000000
 WAL FILE: 0000000100000000000000E3
 WAL FILE: 0000000100000000000000E2

INFO:  2 WAL file(s) will be removed
SERVER NAME: ppas94
BACKUP ID: 1428422324880
BACKUP STATUS: obsolete
BACKUP TIME: 2015-04-07 11:58:45 EDT
BACKUP SIZE: 54.53 MB
WAL FILE(s): 2
 WAL FILE: 0000000100000000000000E1
 WAL FILE: 0000000100000000000000E0
```

The `STATUS` field of the `SHOW-BACKUPS` subcommand displays the current status:

```
-bash-4.1$ bart SHOW-BACKUPS -s ppas94
```

```
 SERVER NAME   BACKUP ID       BACKUP TIME              BACKUP SIZE   WAL(s) SIZE
WAL FILES   STATUS

 ppas94        1428768344061   2015-04-11 12:05:46 EDT   5.72 MB       48.00 MB      3
active
 ppas94        1428684537299   2015-04-10 12:49:00 EDT   5.72 MB       272.00 MB     17
active
 ppas94        1428589759899   2015-04-09 10:29:27 EDT   5.65 MB       96.00 MB      6
active
 ppas94        1428502049836   2015-04-08 10:07:30 EDT   55.25 MB      96.00 MB      6
obsolete
 ppas94        1428422324880   2015-04-07 11:58:45 EDT   54.53 MB      32.00 MB      2
obsolete
 ppas94        1428355371389   2015-04-06 17:22:53 EDT   5.71 MB       16.00 MB      1
keep
```

The details of an individual backup can be displayed using the `SHOW-BACKUPS` subcommand with the `-t` option. Note the status in the `BACKUP STATUS` field.

```
-bash-4.1$ bart SHOW-BACKUPS -s ppas94 -i 1428502049836 -t
SERVER NAME    : ppas94
BACKUP ID      : 1428502049836
BACKUP NAME    : none
BACKUP STATUS  : obsolete
BACKUP TIME    : 2015-04-08 10:07:30 EDT
BACKUP SIZE    : 55.25 MB
WAL(S) SIZE    : 96.00 MB
NO. OF WALS    : 6
FIRST WAL FILE : 0000000100000000000000E2
CREATION TIME  : 2015-04-08 10:07:30 EDT
LAST WAL FILE  : 000000010000000100000003
CREATION TIME  : 2015-04-09 10:25:46 EDT
```

Use the `MANAGE` subcommand with the `-d` option to physically delete the obsolete backups including the unneeded WAL files.

```
-bash-4.1$ bart MANAGE -s ppas94 -d
INFO:  removing all obsolete backups of server 'ppas94'
INFO:  removing obsolete backup '1428502049836'
INFO:  6 WAL file(s) will be removed
INFO:  removing WAL file '000000010000000100000003'
INFO:  removing WAL file '000000010000000100000002'
INFO:  removing WAL file '000000010000000100000001'
INFO:  removing WAL file '000000010000000100000000'
INFO:  removing WAL file '0000000100000000000000E3'
INFO:  removing WAL file '0000000100000000000000E2'
INFO:  removing obsolete backup '1428422324880'
INFO:  2 WAL file(s) will be removed
INFO:  removing WAL file '0000000100000000000000E1'
INFO:  removing WAL file '0000000100000000000000E0'
```

The `SHOW-BACKUPS` subcommand now displays the remaining backups marked as active or keep:

```
-bash-4.1$ bart SHOW-BACKUPS -s ppas94
 SERVER NAME   BACKUP ID       BACKUP TIME              BACKUP SIZE   WAL(s) SIZE
WAL FILES   STATUS

 ppas94        1428768344061   2015-04-11 12:05:46 EDT   5.72 MB       48.00 MB      3
active
```

```
 ppas94        1428684537299   2015-04-10 12:49:00 EDT   5.72 MB       272.00 MB     17
active
 ppas94        1428589759899   2015-04-09 10:29:27 EDT   5.65 MB       96.00 MB      6
active
 ppas94        1428355371389   2015-04-06 17:22:53 EDT   5.71 MB       16.00 MB      1
keep
```

Recovery Window Retention Policy

The following example uses a recovery window retention policy as shown by the following server configuration:

```
[PPAS94_2]
host = 127.0.0.1
port = 5445
user = enterprisedb
archive_command = 'cp %p %a/%f'
retention_policy = 3 DAYS
description = "PPAS 94 server 2"
```

The following is the current set of backups. Note that the last backup in the list has been marked as keep.

```
-bash-4.1$ bart SHOW-BACKUPS -s ppas94_2
 SERVER NAME   BACKUP ID       BACKUP TIME               BACKUP SIZE   WAL(s) SIZE
WAL FILES   STATUS

 ppas94_2      1428933278236   2015-04-13 09:54:40 EDT   5.65 MB       16.00 MB      1
active
 ppas94_2      1428862187757   2015-04-12 14:09:50 EDT   5.65 MB       32.00 MB      2
active
 ppas94 2      1428768351638   2015-04-11 12:05:54 EDT   5.65 MB       32.00 MB      2
active
 ppas94_2      1428684544008   2015-04-10 12:49:06 EDT   5.65 MB       224.00 MB     14
active
 ppas94_2      1428590536488   2015-04-09 10:42:18 EDT   5.65 MB       48.00 MB      3
active
 ppas94_2      1428502171990   2015-04-08 10:09:34 EDT   5.65 MB       80.00 MB      5
keep
```

The current date/time is 2015-04-13 16:46:35 EDT as shown by the following:

```
-bash-4.1$ date
Mon Apr 13 16:46:35 EDT 2015
```

Thus, a 3-day recovery window would evaluate backups prior to 2015-04-10 16:46:35 EDT as obsolete except for those marked as keep.

Invoke the `MANAGE` subcommand with the `-n` option to perform a dry run to observe which active backups would be changed to obsolete according to the retention policy.

```
-bash-4.1$ bart MANAGE -s ppas94_2 -n
INFO:  processing server 'ppas94_2', backup '1428933278236'
INFO:  processing server 'ppas94_2', backup '1428862187757'
INFO:  processing server 'ppas94_2', backup '1428768351638'
INFO:  processing server 'ppas94_2', backup '1428684544008'
INFO:  marking backup '1428684544008' as obsolete
INFO:  14 WAL file(s) marked obsolete
```

```
INFO:  1 Unused WAL file(s) present
INFO:  processing server 'ppas94_2', backup '1428590536488'
INFO:  marking backup '1428590536488' as obsolete
INFO:  3 WAL file(s) marked obsolete
INFO:  1 Unused WAL file(s) present
INFO:  processing server 'ppas94_2', backup '1428502171990'
```

The dry run shows that backups `1428684544008` and `1428590536488` would be marked as obsolete.

Also note that a dry run does not change the backup status. The two backups that would be considered obsolete are still marked as active:

```
-bash-4.1$ bart SHOW-BACKUPS -s ppas94 2
 SERVER NAME   BACKUP ID       BACKUP TIME               BACKUP SIZE   WAL(s) SIZE
WAL FILES   STATUS

 ppas94_2      1428933278236   2015-04-13 09:54:40 EDT   5.65 MB       16.00 MB      1
active
 ppas94 2      1428862187757   2015-04-12 14:09:50 EDT   5.65 MB       32.00 MB      2
active
 ppas94 2      1428768351638   2015-04-11 12:05:54 EDT   5.65 MB       32.00 MB      2
active
 ppas94_2      1428684544008   2015-04-10 12:49:06 EDT   5.65 MB       224.00 MB     14
active
 ppas94 2      1428590536488   2015-04-09 10:42:18 EDT   5.65 MB       48.00 MB      3
active
 ppas94 2      1428502171990   2015-04-08 10:09:34 EDT   5.65 MB       80.00 MB      5
keep
```

Invoke the `MANAGE` subcommand omitting the `-n` option to change and mark the status of the backups as obsolete:

```
-bash-4.1$ bart MANAGE -s ppas94_2
INFO:  processing server 'ppas94_2', backup '1428933278236'
INFO:  processing server 'ppas94_2', backup '1428862187757'
INFO:  processing server 'ppas94_2', backup '1428768351638'
INFO:  processing server 'ppas94_2', backup '1428684544008'
INFO:  marking backup '1428684544008' as obsolete
INFO:  14 WAL file(s) marked obsolete
INFO:  1 Unused WAL file(s) present
INFO:  processing server 'ppas94_2', backup '1428590536488'
INFO:  marking backup '1428590536488' as obsolete
INFO:  3 WAL file(s) marked obsolete
INFO:  1 Unused WAL file(s) present
INFO:  processing server 'ppas94_2', backup '1428502171990'
```

The obsolete backups can be observed in a number of ways. Use the `MANAGE` subcommand with the `-l` option to list the obsolete backups:

```
-bash-4.1$ bart MANAGE -s ppas94_2 -l
INFO:  14 WAL file(s) will be removed
INFO:  1 Unused WAL file(s) will be removed
SERVER NAME: ppas94_2
BACKUP ID: 1428684544008
BACKUP STATUS: obsolete
BACKUP TIME: 2015-04-10 12:49:06 EDT
BACKUP SIZE: 5.65 MB
WAL FILE(s): 14
```

```
UNUSED WAL FILE(s): 1
 WAL FILE: 0000000100000000000002E
 WAL FILE: 0000000100000000000002D
 WAL FILE: 0000000100000000000002C
 WAL FILE: 0000000100000000000002B
 WAL FILE: 0000000100000000000002A
 WAL FILE: 000000010000000000000029
 WAL FILE: 000000010000000000000028
 WAL FILE: 000000010000000000000027
 WAL FILE: 000000010000000000000026
 WAL FILE: 000000010000000000000025
 WAL FILE: 000000010000000000000024
 WAL FILE: 000000010000000000000023
 WAL FILE: 000000010000000000000022
 WAL FILE: 000000010000000000000021
 UNUSED WAL FILE: 0000000100000000000000F.00000028

INFO:  3 WAL file(s) will be removed
INFO:  1 Unused WAL file(s) will be removed
SERVER NAME: ppas94_2
BACKUP ID: 1428590536488
BACKUP STATUS: obsolete
BACKUP TIME: 2015-04-09 10:42:18 EDT
BACKUP SIZE: 5.65 MB
WAL FILE(s): 3
UNUSED WAL FILE(s): 1
 WAL FILE: 000000010000000000000020
 WAL FILE: 00000001000000000000001F
 WAL FILE: 00000001000000000000001E
 UNUSED WAL FILE: 0000000100000000000000F.00000028
```

The `STATUS` field of the `SHOW-BACKUPS` subcommand displays the current status:

```
-bash-4.1$ bart SHOW-BACKUPS -s ppas94_2
 SERVER NAME   BACKUP ID       BACKUP TIME               BACKUP SIZE   WAL(s) SIZE
WAL FILES   STATUS

 ppas94_2      1428933278236   2015-04-13 09:54:40 EDT   5.65 MB       16.00 MB      1
active
 ppas94_2      1428862187757   2015-04-12 14:09:50 EDT   5.65 MB       32.00 MB      2
active
 ppas94_2      1428768351638   2015-04-11 12:05:54 EDT   5.65 MB       32.00 MB      2
active
 ppas94_2      1428684544008   2015-04-10 12:49:06 EDT   5.65 MB       224.00 MB     14
obsolete
 ppas94_2      1428590536488   2015-04-09 10:42:18 EDT   5.65 MB       48.00 MB      3
obsolete
 ppas94_2      1428502171990   2015-04-08 10:09:34 EDT   5.65 MB       80.00 MB      5
keep
```

The details of an individual backup can be displayed using the `SHOW-BACKUPS` subcommand with the `-t` option. Note the status in the `BACKUP STATUS` field.

```
-bash-4.1$ bart SHOW-BACKUPS -s ppas94_2 -i 1428684544008 -t
SERVER NAME    : ppas94_2
BACKUP ID      : 1428684544008
BACKUP NAME    : none
BACKUP STATUS  : obsolete
BACKUP TIME    : 2015-04-10 12:49:06 EDT
BACKUP SIZE    : 5.65 MB
WAL(S) SIZE    : 224.00 MB
NO. OF WALS    : 14
```

```
FIRST WAL FILE : 000000010000000000000021
CREATION TIME  : 2015-04-10 12:49:06 EDT
LAST WAL FILE  : 000000010000000000000002E
CREATION TIME  : 2015-04-11 12:02:15 EDT
```

Use the MANAGE subcommand with the -d option to physically delete the obsolete backups including the unneeded WAL files.

```
-bash-4.1$ bart MANAGE -s ppas94_2 -d
INFO:  removing all obsolete backups of server 'ppas94_2'
INFO:  removing obsolete backup '1428684544008'
INFO:  14 WAL file(s) will be removed
INFO:  1 Unused WAL file(s) will be removed
INFO:  removing WAL file '000000010000000000000002E'
INFO:  removing WAL file '000000010000000000000002D'
INFO:  removing WAL file '000000010000000000000002C'
INFO:  removing WAL file '000000010000000000000002B'
INFO:  removing WAL file '000000010000000000000002A'
INFO:  removing WAL file '000000010000000000000029'
INFO:  removing WAL file '000000010000000000000028'
INFO:  removing WAL file '000000010000000000000027'
INFO:  removing WAL file '000000010000000000000026'
INFO:  removing WAL file '000000010000000000000025'
INFO:  removing WAL file '000000010000000000000024'
INFO:  removing WAL file '000000010000000000000023'
INFO:  removing WAL file '000000010000000000000022'
INFO:  removing WAL file '000000010000000000000021'
INFO:  removing (unused) WAL file '000000010000000000000000F.00000028'
INFO:  removing obsolete backup '1428590536488'
INFO:  3 WAL file(s) will be removed
INFO:  removing WAL file '000000010000000000000020'
INFO:  removing WAL file '00000001000000000000001F'
INFO:  removing WAL file '00000001000000000000001E'
```

The SHOW-BACKUPS subcommand now displays the remaining backups marked as active or keep:

```
-bash-4.1$ bart SHOW-BACKUPS -s ppas94_2
 SERVER NAME   BACKUP ID       BACKUP TIME               BACKUP SIZE   WAL(s) SIZE
WAL FILES   STATUS

 ppas94_2      1428933278236   2015-04-13 09:54:40 EDT   5.65 MB       16.00 MB      1
active
 ppas94_2      1428862187757   2015-04-12 14:09:50 EDT   5.65 MB       32.00 MB      2
active
 ppas94_2      1428768351638   2015-04-11 12:05:54 EDT   5.65 MB       32.00 MB      2
active
 ppas94 2      1428502171990   2015-04-08 10:09:34 EDT   5.65 MB       80.00 MB      5
keep
```

5.3 Basic BART Subcommand Usage

BART is used by invoking the bart program located in the *BART_HOME*/bin directory with the desired subcommand and any applicable subcommand options. The subcommands available with BART are the following:

- **INIT.** Create the BART backup catalog directories, set the Postgres `archive_command` parameter from the setting of the BART `archive_command` parameter for managed database servers of version 9.4, and rebuild the backupinfo files. See Section 5.4.1.
- **BACKUP.** Take a base backup. See Section 5.4.2.
- **SHOW-SERVERS.** Display the database servers managed by BART. See Section 5.4.3.
- **SHOW-BACKUPS.** Display information for the base backups taken by BART. See Section 5.4.4.
- **VERIFY-CHKSUM.** Verify the checksums on the base backups. See Section 5.4.5.
- **MANAGE.** Manage backups using the retention policy. Compress the archived WAL files. See Section 5.4.6.
- **RESTORE.** Restore a base backup and optionally, generate a `recovery.conf` file for restoring archived WAL files for point-in-time recovery. See Section 5.4.7.
- **DELETE.** Delete a base backup. See Section 5.4.8.

There are a number of general BART options that can be given immediately following specification of the `bart` program.

Following the general BART options, if any are given, you can specify a BART subcommand. When invoking a subcommand, the subcommand name is case-insensitive (that is, the subcommand can be specified in uppercase, lowercase, or mixed case).

Each subcommand has a number of its own applicable options that are specified following the subcommand. All options are available in both single-character and multi-character forms.

The option keywords must generally be in lowercase except when specified differently in this chapter.

The general syntax for invoking BART is the following:

```
bart [ gen_option ]... [ subcmd ] [ subcmd_option ]...
```

When invoking BART, the current user must be the BART user account as described in Step 2 of Section 4.1.

If the *BART_HOME*/bin directory is not included in the BART user account's `PATH` environment variable, then *BART_HOME*/bin must be the current working directory when invoking BART. The general invocation syntax would appear as follows:

```
cd BART_HOME/bin
./bart [ gen_option ]... [ subcmd ] [ subcmd_option ]...
```


The BART user account's `LD_LIBRARY_PATH` environment variable must include the directory containing the `libpq` library. This directory is *`POSTGRES_INSTALL_HOME`*`/lib` as shown by the following example:

```
export LD_LIBRARY_PATH=/opt/PostgresPlus/9.4AS/lib/:$LD_LIBRARY_PATH
```

If `LD_LIBRARY_PATH` is not properly set, execution of BART subcommands may fail with the following error message:

```
./bart: symbol lookup error: ./bart: undefined symbol: PQping
```

It is suggested that the `PATH` and the `LD_LIBRARY_PATH` environment variable settings be placed in the BART user account's profile. See Step 3 in Section 4.1.

The following are the general BART options denoted as *`gen_option`* in the preceding syntax diagrams.

Options

`-h`, `--help`

Displays general syntax and information on BART usage.

`-v`, `--version`

Displays the BART version information.

`-d`, `--debug`

Displays debugging output while executing BART subcommands.

`-c`, `--config-path` *`config_file_path`*

Specifies *`config_file_path`* as the full directory path to a BART configuration file. Use this option if you do not want to use the default BART configuration file *`BART_HOME`*`/etc/bart.cfg`.

The following examples illustrate the previously described ways for invoking BART. The BART user account is `bartuser` in these examples.

```
$ su bartuser
Password:
$ export LD_LIBRARY_PATH=/opt/PostgresPlus/9.4AS/lib/:$LD_LIBRARY_PATH
$ cd /usr/edb-bart-1.1/bin
$ ./bart SHOW-SERVERS
```

Run BART from any current working directory:

```
$ su bartuser
Password:
$ export LD_LIBRARY_PATH=/opt/PostgresPlus/9.4AS/lib/:$LD_LIBRARY_PATH
$ export PATH=/usr/edb-bart-1.1/bin:$PATH
$ bart SHOW-SERVERS
```

Use a BART configuration file other than *BART_HOME*/etc/bart.cfg:

```
$ su bartuser
Password:
$ export LD_LIBRARY_PATH=/opt/PostgresPlus/9.4AS/lib/:$LD_LIBRARY_PATH
$ export PATH=/usr/edb-bart-1.1/bin:$PATH
$ bart -c /home/bartuser/bart.cfg SHOW-SERVERS
```

The following section describes the BART subcommands.

5.4 BART Subcommands

This section describes the syntax and usage of the BART subcommands.

All subcommands support a help option (-h, --help). If the help option is specified, information is displayed regarding that particular subcommand. The subcommand, itself, is not executed.

The following is an example of the help option for the BACKUP subcommand:

```
-bash-4.1$ bart BACKUP --help
bart: backup and recovery tool

Usage:
 bart BACKUP [OPTION]...

Options:
  -h, --help           Show this help message and exit
  -s, --server         Name of the server or 'all' to specify all servers
  -F, --format=p|t     Backup output format (tar (default) or plain)
  -z, --gzip           Enables gzip compression of tar files
  -c, --compress-level Specifies the compression level (1 through 9, 9 being
best compression)

  --backup-name        Specify a friendly name for the current backup
```

Note: In the following sections, the help option is omitted from the syntax diagrams for the purpose of providing clarity for the subcommand options.

For clarity, the syntax diagrams also show only the single-character form of the option. The **Options** subsections list both the single-character and multi-character forms of the options.

5.4.1 INIT

The INIT subcommand creates the BART backup catalog directory. For managed Postgres database servers of version 9.4, INIT sets the Postgres archive_command configuration parameter based upon the setting of the BART archive_command parameter in the server section of the BART configuration file. The INIT subcommand also rebuilds the BART backupinfo file.

```
bart INIT [ –s { server_name | all } ] [ -o ]
  [ -r [ -i { backup_id | backup_name | all } ] ]
```

Note: Do not invoke the INIT subcommand while the BART BACKUP subcommand is in progress. Base backups affected by the backup process will be skipped and ignored by the INIT subcommand.

When the INIT subcommand is invoked, several different actions may be performed simultaneously. The following summarizes the actions performed under certain conditions and options.

When the INIT subcommand is invoked, the following action is always performed:

- For the server specified by the -s option, or for all servers if -s all is specified or the -s option is omitted, the BART backup catalog directory structure is completed down to the archived_wals subdirectory if it does not already exist.

When the INIT subcommand is invoked without the -r option, the following action is always performed:

- For the server specified by the -s option, or for all servers if -s all is specified or the -s option is omitted, an attempt is made to set the Postgres archive_command configuration parameter for database server version 9.4. If the archive_command parameter is already set (in other words, archive_command is set to a command string in either the postgresql.conf file or the postgresql.auto.conf file), then the existing archive_command setting is not replaced with the BART archive_command setting unless the -o option is specified as well. See Section 4.2.3.2 for additional information.

When the INIT subcommand is invoked with the -r option, the following action is always performed:

- For the server specified by the -s option, or for all servers if -s all is specified or the -s option is omitted, the backupinfo file is recreated for all backups. If the -i option is included, then the backupinfo file is recreated for the specified backup.

The BART backupinfo file named `backupinfo` is initially created by the `BACKUP` subcommand and contains the backup information used by BART.

When `INIT -r` is invoked, BART rebuilds the backupinfo file using the content of the backup directory.

Note: If the backup was initially created with a user-defined backup name, and then the `INIT -r` option rebuilds that backupinfo file, the user-defined backup name is no longer available. Thus, future references to the backup must use the integer backup identifier.

The following shows the location of the backupinfo file in a backup subdirectory:

```
[root@localhost ppas94]# pwd
/opt/backup/ppas94
[root@localhost ppas94]# ls -l
total 8
drwx------ 2 enterprisedb enterprisedb 4096 Apr  6 11:15 1428333300499
drwx------ 2 enterprisedb enterprisedb 4096 Apr  6 11:15 archived_wals
[root@localhost ppas94]# ls -l 1428333300499
total 56556
-rw-rw-r-- 1 enterprisedb enterprisedb      693 Apr  6 11:15 backupinfo
-rw-rw-r-- 1 enterprisedb enterprisedb 57906176 Apr  6 11:15 base.tar
```

The following is the content of the backupinfo file:

```
BACKUP DETAILS:
BACKUP STATUS: active
BACKUP IDENTIFIER: 1428333300499
BACKUP NAME: none
BACKUP LOCATION: /opt/backup/ppas94/1428333300499
BACKUP SIZE: 55.22 MB
BACKUP FORMAT: tar
BACKUP TIMEZONE: US/Eastern
XLOG METHOD: fetch
BACKUP CHECKSUM(s): 1
 ChkSum                           File
 53122246ac06061b291595daa1ca9650   base.tar

TABLESPACE(s): 0
START WAL LOCATION: 0/93000028 (file 000000010000000000000093)
STOP WAL LOCATION: 0/930000B8 (file 000000010000000000000093)
CHECKPOINT LOCATION: 0/93000028
BACKUP METHOD: streamed
BACKUP FROM: master
START TIME: 2015-04-06 11:15:00 EDT
LABEL: pg_basebackup base backup
STOP TIME: 2015-04-06 11:15:01 EDT

TOTAL DURATION: 1 sec(s)
```

If the backupinfo file is missing, then the following error message appears when a BART subcommand is invoked:

```
-bash-4.1$ bart SHOW-BACKUPS
ERROR: 'backupinfo' file does not exists for backup '1428333300499'
please use 'INIT -r' to generate the file
```

The backupinfo file could be missing if the BACKUP subcommand did not complete successfully.

Note: The backupinfo file was not created by, nor used by BART version 1.0.x. Thus, if you wish to use BART 1.1 to access backups that were created with previous versions of BART, use the INIT -r option to create the backupinfo files. See Section 3.3 for information on upgrading from earlier BART versions.

Options

-s, --server { *server_name* | all }

server_name is the name of the database server to which the INIT actions are to be applied. If all is specified or if the option is omitted, the actions are applied to all servers.

-o, --override

Override the existing, active Postgres archive_command configuration parameter setting in the postgresql.conf file or the postgresql.auto.conf file using the BART archive_command parameter in the BART configuration file. The INIT generated archive command string is written to the postgresql.auto.conf file. This option only applies for database server version 9.4. The replication database user of these database servers must be a superuser. **Note:** If the archive_mode configuration parameter is set to off, then the -o option must be used to set the Postgres archive_command using the BART archive_command parameter in the BART configuration file even if the archive_command is not currently set in postgresql.conf nor in postgresql.auto.conf.

-r, --rebuild

Rebuilds the backupinfo file. This file is a text file named backupinfo located in each base backup subdirectory.

-i, --backupid { *backup_id* | *backup_name* | all }

backup_id is an integer, base backup identifier. *backup_name* is the user-defined alphanumeric name for the base backup. If all is specified or if the option is omitted, the backupinfo files of all backups for the database servers specified by the -s option are recreated. The -i option can only be used with the -r option.

Example 1 (archive_command)

The following example completes the BART backup catalog directory and sets the Postgres `archive_command` using the default BART archive command format of `scp %p %h:%a/%f`. The default BART archive command format is used when the BART `archive_command` parameter is not explicitly included or assigned a command string within the server section of the BART configuration file.

The following is the BART configuration file:

```
[BART]
bart-host= bartuser@192.168.2.22
backup_path = /opt/backup
pg_basebackup_path = /opt/PostgresPlus/9.4AS/bin/pg_basebackup
logfile = /tmp/bart.log

[PPAS94]
host = 127.0.0.1
port = 5444
user = repuser
description = "PPAS 94 server"
```

The `archive_mode` and `archive_command` parameters in the database server are set as follows:

```
edb=# SHOW archive_mode;
 archive_mode
--------------
 on
(1 row)

edb=# SHOW archive_command;
 archive_command
-----------------

(1 row)
```

The `INIT` subcommand is invoked. The `-o` option is not necessary since `archive_mode` is `on` and `archive_command` is not set.

```
[bartuser@localhost ~]$ bart INIT
INFO:  setting archive_command for server 'ppas94'
WARNING: archive_command is set. server restart is required
```

The BART backup catalog directory structure is completed:

```
[root@localhost opt]# pwd
/opt
[root@localhost opt]# ls -l backup
total 4
drwx------ 3 bartuser bartuser 4096 Apr  3 16:44 ppas94
[root@localhost opt]# ls -l backup/ppas94
total 4
drwx------ 2 bartuser bartuser 4096 Apr  3 16:44 archived_wals
```

Reload the database server configuration. (Restart of the database server is not necessary to reset only the `archive_command` parameter.)

```
 [root@localhost opt]# /etc/init.d/ppas-9.4 reload
Reloading Postgres Plus Advanced Server 9.4:
server signaled
```

The Postgres `archive_command` is now set as follows:

```
edb=# SHOW archive_command;
                          archive_command
------------------------------------------------------------------
 scp %p bartuser@192.168.2.22:/opt/backup/ppas94/archived_wals/%f
(1 row)
```

The new command string is written to the `postgresql.auto.conf` file:

```
# Do not edit this file manually!
# It will be overwritten by ALTER SYSTEM command.
archive_command = 'scp %p bartuser@192.168.2.22:/opt/backup/ppas94/archived_wals/%f'
```

Example 2 (archive_command)

The following example overrides an existing, active, archive command by resetting the Postgres `archive_command` parameter from the BART `archive_command = 'cp %p %a/%f'` parameter in the BART configuration file.

The following is the BART configuration file:

```
[BART]
bart-host= enterprisedb@192.168.2.22
backup_path = /opt/backup_edb
pg_basebackup_path = /opt/PostgresPlus/9.4AS/bin/pg_basebackup
logfile = /tmp/bart.log

[PPAS94]
host = 127.0.0.1
port = 5444
user = repuser
archive_command = 'cp %p %a/%f'
description = "PPAS 94 server"
```

The `archive_mode` and `archive_command` parameters in the database server are set as follows:

```
edb=# SHOW archive_mode;
 archive_mode
--------------
 on
(1 row)

edb=# SHOW archive_command;
                          archive_command
------------------------------------------------------------------
 scp %p bartuser@192.168.2.22:/opt/backup/ppas94/archived_wals/%f
(1 row)
```

The `INIT` subcommand is invoked with the `-o` option to override the current Postgres `archive_command` setting.

```
-bash-4.1$ bart INIT -s ppas94 -o
INFO:  setting archive_command for server 'ppas94'
WARNING: archive_command is set. server restart is required
```

Reload the database server configuration. (Restart of the database server is not necessary to reset only the `archive_command` parameter.)

```
[root@localhost tmp]# /etc/init.d/ppas-9.4 reload
Reloading Postgres Plus Advanced Server 9.4:
server signaled
```

The Postgres `archive_command` is now set as follows:

```
edb=# SHOW archive_command;
                archive_command
-----------------------------------------------
 cp %p /opt/backup_edb/ppas94/archived_wals/%f
(1 row)
```

The new command string is written to the `postgresql.auto.conf` file:

```
# Do not edit this file manually!
# It will be overwritten by ALTER SYSTEM command.
archive_command = 'cp %p /opt/backup_edb/ppas94/archived_wals/%f'
```

Example (rebuild backupinfo option)

The following example rebuilds the backupinfo file of the specified backup for database server `ppas94`.

```
-bash-4.1$ bart INIT -s ppas94 -r -i 1428346620427
INFO:  rebuilding BACKUPINFO for backup '1428346620427' of server 'ppas94'
INFO:  backup checksum: ced59b72a7846ff8fb8afb6922c70649 of base.tar
```

The following example rebuilds the backupinfo files of all backups for all database servers.

```
-bash-4.1$ bart INIT -r
INFO:  rebuilding BACKUPINFO for backup '1428347191544' of server 'ppas94'
INFO:  backup checksum: 1ac5c61f055c910db314783212f2544f of base.tar
INFO:  rebuilding BACKUPINFO for backup '1428346620427' of server 'ppas94'
INFO:  backup checksum: ced59b72a7846ff8fb8afb6922c70649 of base.tar
INFO:  rebuilding BACKUPINFO for backup '1428347198335' of server 'ppas94_2'
INFO:  backup checksum: a8890dd8ab7e6be5d5bc0f38028a237b of base.tar
INFO:  rebuilding BACKUPINFO for backup '1428346957515' of server 'ppas94_2'
INFO:  backup checksum: ea62549cf090573625d4adeb7d919700 of base.tar
```

5.4.2 BACKUP

The BACKUP subcommand creates a base backup by invoking the Postgres pg_basebackup utility program.

Note: Use the --debug option prior to the BACKUP subcommand to see the exact invocation of pg_basebackup.

```
bart BACKUP -s { server_name | all } [ -F { p | t } ]
  [ -z ] [ -c compression_level ]
  [ --backup-name backup_name ]
```

By default, the target format is a tar file, and the -x flag is passed to pg_basebackup to ensure that the WAL files are also archived.

The backup is saved in the directory formed by *backup_path*/*server_name*/*backup_id* where *backup_path* is the value assigned to the backup_path parameter in the BART configuration file, *server_name* is the lowercase name of the database server as listed in the configuration file, and *backup_id* is an integer, base backup identifier assigned by BART to the particular base backup.

MD5 checksums of the base backup and any user-defined tablespaces are saved as well for tar backups.

While a BACKUP subcommand is in progress, no other processes are allowed to access or interfere with the affected base backups. Therefore any of the following subcommands issued while a backup is in progress will skip and ignore those affected base backups – INIT, SHOW-BACKUPS, VERIFY-CHKSUM, MANAGE, and DELETE.

Before performing the backup, BART checks to ensure that there is enough disk space to completely store the backup in the BART backup catalog. If BART detects that there is not enough disk space, then an error message is displayed as shown by the following:

```
edb@localhost bin]$ ./bart backup -s ppas93 -Ft

WARNING: xlog-method is empty, defaulting to global policy
ERROR: backup failed for server 'ppas93'
free disk space is not enough to backup the server 'ppas93'
space available 13.35 GB, approximately required 14.65 GB
```

No backup files are copied to the BART backup catalog when this shortage of disk space is detected.

Note: Although this capability to check and warn if there is not enough disk space available before copying backup files is provided with the BACKUP subcommand, the RESTORE subcommand does not have this same capability. Thus, it is possible that the

RESTORE subcommand may result in an error while copying files if there is not enough disk space available.

In the postgresql.conf file, be sure the wal_keep_segments configuration parameter is set to a sufficiently large value, otherwise you may encounter the following error from pg_basebackup during usage of the BACKUP subcommand:

```
ERROR: backup failed for server 'ppas93'
command failed with exit code 1
pg_basebackup: could not get transaction log end position from server: ERROR:
requested WAL segment 00000001000000D50000006B has already been removed
```

A low setting of the wal_keep_segments configuration parameter may result in the deletion of some WAL files before pg_basebackup has had a chance to save them to the BART backup catalog.

For information about the wal_keep_segments parameter, see the *PostgreSQL Core Documentation* available at:

http://www.postgresql.org/docs/9.5/static/runtime-config-replication.html

If in the BART configuration file, parameter setting xlog-method=stream applies to a given database server, streaming of the transaction log in parallel with creation of the backup is performed for that database server, otherwise the transaction log files are collected upon completion of the backup. See Section 4.1 for global setting of xlog-method. See Section 4.2.5 for setting of xlog-method by database server.

Note: If the transaction log streaming method is used, the -F p option for a plain text backup format must be specified with the BACKUP subcommand.

Note: If the database server contains user-defined tablespaces (that is, tablespaces created with the CREATE TABLESPACE command), only the tar backup file format is supported. In other words, the BACKUP subcommand option -F p for specifying a plain text backup file format may not be used. Consequently, transaction log streaming with the BACKUP subcommand cannot be used. Thus, the xlog-method = fetch parameter setting must be in effect for such database servers containing user-defined tablespaces.

Options

-s, --server { *server_name* | all }

server_name is the name of the database server to be backed up as specified in the BART configuration file. If all is specified, all servers are backed up. **Note:** If all is specified, and a connection to a database server listed in the BART configuration file cannot be opened, the backup for that database server is

skipped, but the backup operation continues for the other database servers. The following error message is displayed when a database server connection fails:

```
ERROR: backup failed for server 'ppas93'
connection to the server failed: could not connect to server: Connection refused
    Is the server running on host "172.16.114.132" and accepting
    TCP/IP connections on port 5444?
```

`-F, --format { p | t }`

Specifies the backup file format. Use `p` for plain text or `t` for tar. If the option is omitted, the default is tar format.

`-z, --gzip`

Specifies usage of gzip compression on the tar file output using the default compression level. The default compression level is typically 6. This option is applicable only for the tar format.

`-c, --compress-level` *`compression_level`*

Specifies the gzip compression level on the tar file output. *`compression_level`* is a digit from 1 through 9, inclusive, with 9 being the best compression. This option is applicable only for the tar format.

`--backup-name` *`backup_name`*

User-defined, friendly name to be assigned to the base backup. This is an alphanumeric string that may include the following variables to be substituted by the timestamp values when the backup is taken: 1) `%year` – 4-digit year, 2) `%month` – 2-digit month, 3) `%day` – 2-digit day, 4) `%hour` – 2-digit hour, 5) `%minute` – 2-digit minute, and 6) `%second` – 2-digit second. To include the percent sign (`%`) as a character in the backup name, specify `%%` in the alphanumeric string. Enclose the string in single quotes (`'`) or double quotes (`"`) if it contains space characters. Use of space characters, however, then requires enclosing the backup name in quotes when referenced with the `-i` option by other subcommands. This option overrides the `backup-name` parameter in the server section of the BART configuration file. If the `--backup-name` option is not specified, and the `backup-name` parameter is not set for this database server in the BART configuration file, then the base backup can only be referenced in other BART subcommands by the BART assigned, integer backup identifier.

Example

The following example creates a base backup of server `ppas93_remote` in the default tar format with gzip compression. The debug option (`--debug`) is also specified so the

actual `pg_basebackup` command is shown. Note that checksums are generated for the base backup and user-defined tablespaces for the tar format backup.

```
-bash-4.1$ bart --debug BACKUP -s ppas93_remote -z
DEBUG: Server: ppas94, No. Backups 8
DEBUG: Server: pg94, Now: 2015-04-21 17:16:51 EDT, RetentionWindow: 345600 (secs) ==>
96 hour(s)

DEBUG: Exec Command: /opt/PostgresPlus/9.4AS/bin/pg_basebackup --version
INFO:  creating backup for server 'ppas93_remote'
INFO:  backup identifier: '1429651011080'
DEBUG: internal Backup Command to be execute:
'/opt/PostgresPlus/9.4AS/bin/pg_basebackup -D /opt/backup/ppas93_remote/1429651011080 -
X fetch -P  -Ft -z -d "host=192.168.2.24 port=5444 user=repuser"'
54699/54699 kB (100%), 3/3 tablespaces

INFO:  backup completed successfully
DEBUG: Exec Command: tar  -C /opt/backup/ppas93_remote/1429651011080 -xzf
/opt/backup/ppas93_remote/1429651011080/base.tar.gz backup_label
DEBUG: calculate checksum for backup '/opt/backup/ppas93 remote/1429651011080'
DEBUG: calculating checksum of file
'/opt/backup/ppas93_remote/1429651011080/base.tar.gz'
INFO:  backup checksum: c02ed03588042c9a74f1a6483b271f18 of base.tar.gz
DEBUG: calculating checksum of file
'/opt/backup/ppas93 remote/1429651011080/16644.tar.gz'
INFO:  backup checksum: 94a31dbe8cdf749b47d2f73eeee3ace9 of 16644.tar.gz
DEBUG: calculating checksum of file
'/opt/backup/ppas93_remote/1429651011080/16645.tar.gz'
INFO:  backup checksum: 4959cf01c588844b505444192f637a6b of 16645.tar.gz
DEBUG: start time: 1429654611, stop time: 1429654614, duration: 3
DEBUG: Backup Info file created at '/opt/backup/ppas93 remote/1429651011080/backupinfo'
INFO:
BACKUP DETAILS:
BACKUP STATUS: active
BACKUP IDENTIFIER: 1429651011080
BACKUP NAME: none
BACKUP LOCATION: /opt/backup/ppas93 remote/1429651011080
BACKUP SIZE: 6.41 MB
BACKUP FORMAT: tar.gz
XLOG METHOD: fetch
BACKUP CHECKSUM(s): 3
 ChkSum                           File
 c02ed03588042c9a74f1a6483b271f18   base.tar.gz
 94a31dbe8cdf749b47d2f73eeee3ace9   16644.tar.gz
 4959cf01c588844b505444192f637a6b   16645.tar.gz

TABLESPACE(s): 2
 Oid     Name       Location
 16644   tblspc_1   /mnt/tablespace_1
 16645   tblspc_2   /mnt/tablespace_2

START WAL LOCATION: 000000010000000000000000B
BACKUP METHOD: streamed
BACKUP FROM: master
START TIME: 2015-04-21 17:16:51 EDT
STOP TIME: 2015-04-21 17:16:54 EDT
TOTAL DURATION: 3 sec(s)
```

The following shows the directory containing the base backup:

```
-bash-4.1$ pwd
/opt/backup
-bash-4.1$ ls -l ppas93_remote
total 8
drwx------ 2 enterprisedb enterprisedb 4096 Apr 21 17:16 1429651011080
drwx------ 2 enterprisedb enterprisedb 4096 Apr 21 17:16 archived_wals
```

The following example shows the creation of a base backup while streaming the transaction log.

Note that the `-F p` option must be specified with the `BACKUP` subcommand when streaming is used.

```
-bash-4.1$ bart --debug BACKUP -s PPAS94 -F p
DEBUG: Server: ppas94, No. Backups 8
DEBUG: Server: pg94, Now: 2015-04-21 17:14:05 EDT, RetentionWindow: 345600 (secs) ==>
96 hour(s)

DEBUG: Exec Command: /opt/PostgresPlus/9.4AS/bin/pg basebackup --version
INFO:  creating backup for server 'ppas94'
INFO:  backup identifier: '1429650845239'
DEBUG: internal Backup Command to be execute:
'/opt/PostgresPlus/9.4AS/bin/pg basebackup -D /opt/backup/ppas94/1429650845239/base -X
stream -P  -Fp -d "host=127.0.0.1 port=5444 user=enterprisedb"'
40057/40057 kB (100%), 1/1 tablespace

INFO:  backup completed successfully
DEBUG: start time: 1429654445, stop time: 1429654445, duration: 0
DEBUG: Backup Info file created at '/opt/backup/ppas94/1429650845239/backupinfo'
INFO:
BACKUP DETAILS:
BACKUP STATUS: active
BACKUP IDENTIFIER: 1429650845239
BACKUP NAME: ppas94 2015-04-21T17:14:05
BACKUP LOCATION: /opt/backup/ppas94/1429650845239
BACKUP SIZE: 54.41 MB
BACKUP FORMAT: plain
XLOG METHOD: stream
BACKUP CHECKSUM(s): 0
TABLESPACE(s): 0
START WAL LOCATION: 000000010000000100000F2
BACKUP METHOD: streamed
BACKUP FROM: master
START TIME: 2015-04-21 17:14:05 EDT
STOP TIME: 2015-04-21 17:14:05 EDT
TOTAL DURATION: 0 sec(s)
```

The following example shows the assignment of a user-defined backup name with the `--backup-name` option:

```
-bash-4.1$ bart BACKUP -s ppas94 --backup-name ppas94_%year-%month-%day

INFO:  creating backup for server 'ppas94'
INFO:  backup identifier: '1430239348243'
56452/56452 kB (100%), 1/1 tablespace

INFO:  backup completed successfully
INFO:  backup checksum: 41677579ee5dbf8c3a2d42b5fedd886c of base.tar
INFO:
BACKUP DETAILS:
BACKUP STATUS: active
BACKUP IDENTIFIER: 1430239348243
BACKUP NAME: ppas94_2015-04-28
BACKUP LOCATION: /opt/backup/ppas94/1430239348243
BACKUP SIZE: 55.13 MB
BACKUP FORMAT: tar
XLOG METHOD: fetch
BACKUP CHECKSUM(s): 1
 ChkSum                            File
 41677579ee5dbf8c3a2d42b5fedd886c  base.tar
```

```
TABLESPACE(s): 0
START WAL LOCATION: 000000010000000000000004B
BACKUP METHOD: streamed
BACKUP FROM: master
START TIME: 2015-04-28 12:42:28 EDT
STOP TIME: 2015-04-28 12:42:28 EDT
TOTAL DURATION: 0 sec(s)
```

5.4.3 SHOW-SERVERS

The `SHOW-SERVERS` subcommand displays the information for the managed database servers listed in the BART configuration file.

```
bart SHOW-SERVERS [ -s { server_name | all } ]
```

The default action is to show all servers.

Options

`-s, --server { server_name | all }`

`server_name` is the name of the database server whose BART configuration information is to be displayed. If `all` is specified or if the option is omitted, information for all database servers is displayed.

Example

The following example shows all database servers managed by BART:

```
-bash-4.1$ bart SHOW-SERVERS
SERVER NAME         : pg94
HOST NAME           : 127.0.0.1
USER NAME           : postgres
PORT                : 5432
REMOTE HOST         :
RETENTION POLICY    : 2015-04-17 17:25:03 EDT
DISK UTILIZATION    : 64.00 MB
NUMBER OF ARCHIVES  : 0
ARCHIVE PATH        : /opt/backup/pg94/archived_wals
ARCHIVE COMMAND     : scp %p
enterprisedb@192.168.2.22:/opt/backup/pg94/archived_wals/%f
XLOG METHOD         : fetch
WAL COMPRESSION     : disabled
TABLESPACE PATH(s)  :
DESCRIPTION         : "PostgreSQL 94 server"

SERVER NAME         : ppas93_remote
HOST NAME           : 192.168.2.24
USER NAME           : repuser
PORT                : 5444
REMOTE HOST         : enterprisedb@192.168.2.24
```

```
RETENTION POLICY    : none
DISK UTILIZATION    : 38.41 MB
NUMBER OF ARCHIVES  : 3
ARCHIVE PATH        : /opt/backup/ppas93_remote/archived_wals
ARCHIVE COMMAND     : scp %p
enterprisedb@192.168.2.22:/opt/backup/ppas93_remote/archived_wals/%f
XLOG METHOD         : fetch
WAL COMPRESSION     : disabled
TABLESPACE PATH(s)  :
                      16644 = /opt/restore_tblspc_1
                      16645 = /opt/restore_tblspc_2
DESCRIPTION         : "PPAS 93 remote server"

SERVER NAME         : ppas94
BACKUP FRIENDLY NAME: ppas94_%year-%month-%dayT%hour:%minute:%second
HOST NAME           : 127.0.0.1
USER NAME           : enterprisedb
PORT                : 5444
REMOTE HOST         :
RETENTION POLICY    : 8 Backups
DISK UTILIZATION    : 182.42 MB
NUMBER OF ARCHIVES  : 8
ARCHIVE PATH        : /opt/backup/ppas94/archived_wals
ARCHIVE COMMAND     : cp %p /opt/backup/ppas94/archived_wals/%f
XLOG METHOD         : stream
WAL COMPRESSION     : disabled
TABLESPACE PATH(s)  :
DESCRIPTION         : "PPAS 94 server"
```

5.4.4 SHOW-BACKUPS

The `SHOW-BACKUPS` subcommand displays the base backup information for the managed database servers.

```
bart SHOW-BACKUPS [ -s { server_name | all } ]
  [ -i { backup_id | backup_name | all } ]
  [ -t ]
```

If all options are omitted, the default action is to show all backups of all servers with the exception as described by the following note:

Note: If `SHOW-BACKUPS` is invoked while the BART `BACKUP` subcommand is in progress, base backups affected by the backup process show an in progress status in the displayed backup information.

Options

`-s, --server { server_name | all }`

> *`server_name`* is the name of the database server whose base backup information is to be displayed. If `all` is specified or if the option is omitted, the base backup information for all database servers is displayed.

`-i, --backupid { backup_id | backup_name | all }`

> *backup_id* is an integer, base backup identifier. *backup_name* is the user-defined alphanumeric name for the base backup. If `all` is specified or if the option is omitted, all base backup information for the relevant database server is displayed.

`-t, --toggle`

> Display more comprehensive backup information in list format. If omitted, the default is a briefer, tabular format.

Example

The following example shows all base backups:

```
-bash-4.1$ bart SHOW-BACKUPS
 SERVER NAME     BACKUP ID       BACKUP TIME               BACKUP SIZE   WAL(s) SIZE
WAL FILES   STATUS

 pg94            1429651642513   2015-04-21 17:27:24 EDT   2.45 MB       16.00 MB
1           active
 ppas93 remote   1429651011080   2015-04-21 17:16:54 EDT   6.41 MB       48.00 MB
3           active
 ppas94          1429650845239   2015-04-21 17:14:05 EDT   54.41 MB      48.00 MB
3           active
```

The following example shows more detailed information using the `-t` option.

```
-bash-4.1$ bart SHOW-BACKUPS -s ppas94 -i 1429650845239 -t
SERVER NAME    : ppas94
BACKUP ID      : 1429650845239
BACKUP NAME    : ppas94_2015-04-21T17:14:05
BACKUP STATUS  : active
BACKUP TIME    : 2015-04-21 17:14:05 EDT
BACKUP SIZE    : 54.41 MB
WAL(S) SIZE    : 48.00 MB
NO. OF WALS    : 3
FIRST WAL FILE : 0000000100000001000000F2
CREATION TIME  : 2015-04-21 17:14:05 EDT
LAST WAL FILE  : 0000000100000001000000F4
CREATION TIME  : 2015-04-21 17:24:05 EDT
```

5.4.5 VERIFY-CHKSUM

The `VERIFY-CHKSUM` subcommand verifies the MD5 checksums of the base backups and any user-defined tablespaces for the specified database server or for all database servers.

```
bart VERIFY-CHKSUM
  [ -s { server_name | all } ]
```

```
[ -i { backup_id | backup_name | all } ]
```

The checksum is verified by comparing the current checksum of the backup against the checksum when the backup was taken.

The `VERIFY-CHKSUM` subcommand is not applicable to plain format backups. It is only used for tar format backups.

Note: If `VERIFY-CHKSUM` is invoked while the BART `BACKUP` subcommand is in progress, base backups affected by the backup process will be skipped and ignored by the `VERIFY-CHKSUM` subcommand.

Options

`-s, --server { server_name | all }`

> *`server_name`* is the name of the database server whose tar backup checksums are to be verified. If `all` is specified or if the `-s` option is omitted, the checksums are verified for all database servers.

`-i, --backupid { backup_id | backup_name | all }`

> *`backup_id`* is the integer, base backup identifier of a tar format base backup whose checksum is to be verified along with any user-defined tablespaces. *`backup_name`* is the user-defined alphanumeric name for the base backup. If `all` is specified or if the `-i` option is omitted, the checksums of all tar backups for the relevant database server are verified.

Example

The following example verifies the checksum of all tar format backups of the specified database server.

```
-bash-4.1$ bart VERIFY-CHKSUM -s ppas94 -i all
 SERVER NAME   BACKUP ID       VERIFY

 ppas94        1430239348243   OK
 ppas94        1430232284202   OK
 ppas94        1430232016284   OK
 ppas94        1430231949065   OK
 ppas94        1429821844271   OK
```


5.4.6 MANAGE

The MANAGE subcommand evaluates, marks, and deletes backups based upon the retention policy. The MANAGE subcommand also invokes compression on the archived WAL files based upon the wal_compression parameter.

```
bart MANAGE [ -s { server_name | all } ]
  [ -l ] [ -d ]
  [ -c { keep | nokeep }
    -i { backup_id | backup_name | all } ]
  [ -n ]
```

Note: Do not invoke the MANAGE subcommand while the BART BACKUP subcommand is in progress. Base backups affected by the backup process will be skipped and ignored by the MANAGE subcommand.

Application of a retention policy is dependent upon the retention_policy parameter in the BART configuration file. Management of backups such as evaluating the backups according to the retention policy, marking their status, and deleting obsolete backups are then performed with the MANAGE subcommand. See Section 5.2 for information on retention policy management.

WAL compression is controlled by the wal_compression parameter in the BART configuration file. See sections 4.1 and 4.2.5 for information on setting this parameter.

When the MANAGE subcommand is invoked, several different actions may be performed simultaneously. The following summarizes the actions performed under certain conditions and options.

When the MANAGE subcommand is invoked with no options or with only the -s option to specify all or a particular database server, the following actions are performed:

- For the server specified by the -s option, or for all servers if -s all is specified or the -s option is omitted, active backups are marked as obsolete in accordance with the retention policy.
- All backups that were marked obsolete or keep prior to invoking the MANAGE subcommand remain marked with the same prior status.
- If WAL compression is enabled for the database server, then any uncompressed, archived WAL files in the BART backup catalog of the database server are compressed with gzip.

When the MANAGE subcommand is invoked with any other option besides the -s option, the following actions are performed:

- For the server specified by the -s option, or for all servers if -s all is specified or the -s option is omitted, the action performed is determined by the other

specified options (that is, `-l` to list obsolete backups, `-d` to delete obsolete backups, `-c` to keep or to return backups to active status, or `-n` to perform a dry run of any action).

- No marking of active backups to obsolete status is performed regardless of the retention policy.
- All backups that were marked obsolete or keep prior to invoking the `MANAGE` subcommand remain marked with the same prior status unless the `-c` option (without the `-n` option) is specified to change the backup status of the particular backup or all backups referenced with the `-i` option.
- No compression is applied to any uncompressed, archived WAL file in the BART backup catalog regardless of whether or not WAL compression is enabled.

The following are additional considerations when using WAL compression:

- The gzip compression program must be installed on the BART host and be accessible in the `PATH` of the BART user account.
- When the `RESTORE` subcommand is invoked, if the `-c` option is specified or if the enabled setting of the `copy_wals_during_restore` BART configuration parameter is in effect for the database server, then the following actions occur: If compressed, archived WAL files are stored in the BART backup catalog and the location to which the WAL files are to be restored is on a remote host relative to the BART host, then the archived WAL files are transmitted across the network to the remote host in compressed format, but only if the gzip compression program is accessible in the `PATH` of the remote user account that is used to log into the remote host when performing the `RESTORE` operation. This remote user is specified with either the `remote-host` parameter in the BART configuration file (see Section 4.2.5) or the `RESTORE -r` option (see Section 5.4.7). Transmission of compressed WAL files results in less network traffic. After the compressed WAL files are transmitted across the network, the `RESTORE` subcommand uncompresses the files for the point-in-time recovery operation.
- If the gzip program is not accessible on the remote host in the manner described in the previous bullet point, then the compressed, archived WAL files are first uncompressed while on the BART host, then transmitted across the network to the remote host in uncompressed format.
- When the `RESTORE` subcommand is invoked without the `-c` option and the disabled setting of the `copy_wals_during_restore` BART configuration parameter is in effect for the database server, then any compressed, archived WAL files needed for the `RESTORE` operation are uncompressed in the BART backup catalog. The uncompressed WAL files can then be streamed to the remote host by the `restore_command` in the `recovery.conf` file when the database server archive recovery begins.

Options

`-s, --server { server_name | all }`

server_name is the name of the database server to which the actions are to be applied. If `all` is specified or if the `-s` option is omitted, the actions are applied to all database servers.

`-l, --list-obsolete`

List the backups marked as obsolete.

`-d, --delete-obsolete`

Delete the backups marked as obsolete. This action physically deletes the base backup along with its archived WAL files.

`-c, --change-status { keep | nokeep }`

Change the status of a backup to `keep` to retain it indefinitely. Specify `nokeep` to change the status of any backup, regardless of its current marked status, back to active status. The backup can then be re-evaluated and possibly be marked to obsolete according to the retention policy by subsequent usage of the `MANAGE` subcommand. **Note:** The `-i` option must also be specified when using the `-c` option.

`-i, --backupid { backup_id | backup_name | all }`

backup_id is an integer, base backup identifier. *backup_name* is the user-defined alphanumeric name for the base backup. If `all` is specified, the actions are applied to all backups. **Note:** The `-i` option must only be used with the `-c` option.

`-n, --dry-run`

Displays the results as if the operation was performed, however, no changes are actually made. In other words, a test run is performed so that you can see the results prior to actually implementing the actions. Thus, `-n` specified with the `-d` option displays which backups would be deleted, but does not actually delete the backups. Specifying the `-n` option with the `-c` option displays the keep or nokeep action, but does not actually change the backup from its current status. Specifying `-n` alone with no other options, or with only the `-s` option, displays which active backups would be marked as obsolete, but does not actually change the backup status. In addition, no compression is performed on uncompressed, archived WAL files even if WAL compression is enabled for the database server.

Example

The following example performs a dry run for the specified database server displaying which active backups are evaluated as obsolete according to the retention policy, but does not actually change the backup status:

```
-bash-4.1$ bart MANAGE -s ppas94 -n
INFO:  processing server 'ppas94', backup '1428768344061'
INFO:  processing server 'ppas94', backup '1428684537299'
INFO:  processing server 'ppas94', backup '1428589759899'
INFO:  processing server 'ppas94', backup '1428502049836'
INFO:  marking backup '1428502049836' as obsolete
INFO:  6 WAL file(s) marked obsolete
INFO:  processing server 'ppas94', backup '1428422324880'
INFO:  marking backup '1428422324880' as obsolete
INFO:  2 WAL file(s) marked obsolete
INFO:  processing server 'ppas94', backup '1428355371389'
```

The following example marks active backups as obsolete according to the retention policy for the specified database server:

```
-bash-4.1$ bart MANAGE -s ppas94
INFO:  processing server 'ppas94', backup '1428768344061'
INFO:  processing server 'ppas94', backup '1428684537299'
INFO:  processing server 'ppas94', backup '1428589759899'
INFO:  processing server 'ppas94', backup '1428502049836'
INFO:  marking backup '1428502049836' as obsolete
INFO:  6 WAL file(s) marked obsolete
INFO:  processing server 'ppas94', backup '1428422324880'
INFO:  marking backup '1428422324880' as obsolete
INFO:  2 WAL file(s) marked obsolete
INFO:  processing server 'ppas94', backup '1428355371389'
```

The following example lists backups marked as obsolete for the specified database server:

```
-bash-4.1$ bart MANAGE -s ppas94 -l
INFO:  6 WAL file(s) will be removed
SERVER NAME: ppas94
BACKUP ID: 1428502049836
BACKUP STATUS: obsolete
BACKUP TIME: 2015-04-08 10:07:30 EDT
BACKUP SIZE: 55.25 MB
WAL FILE(s): 6
 WAL FILE: 000000010000000100000003
 WAL FILE: 000000010000000100000002
 WAL FILE: 000000010000000100000001
 WAL FILE: 000000010000000100000000
 WAL FILE: 0000000100000000000000E3
 WAL FILE: 0000000100000000000000E2

INFO:  2 WAL file(s) will be removed
SERVER NAME: ppas94
BACKUP ID: 1428422324880
BACKUP STATUS: obsolete
BACKUP TIME: 2015-04-07 11:58:45 EDT
BACKUP SIZE: 54.53 MB
WAL FILE(s): 2
 WAL FILE: 0000000100000000000000E1
 WAL FILE: 0000000100000000000000E0
```

The following example deletes the obsolete backups for the specified database server:

```
-bash-4.1$ bart MANAGE -s ppas94 -d
INFO:  removing all obsolete backups of server 'ppas94'
INFO:  removing obsolete backup '1428502049836'
INFO:  6 WAL file(s) will be removed
INFO:  removing WAL file '000000010000000100000003'
INFO:  removing WAL file '000000010000000100000002'
INFO:  removing WAL file '000000010000000100000001'
INFO:  removing WAL file '000000010000000100000000'
INFO:  removing WAL file '0000000100000000000000E3'
INFO:  removing WAL file '0000000100000000000000E2'
INFO:  removing obsolete backup '1428422324880'
INFO:  2 WAL file(s) will be removed
INFO:  removing WAL file '0000000100000000000000E1'
INFO:  removing WAL file '0000000100000000000000E0'
```

The following example changes the specified backup to keep status to retain it indefinitely:

```
-bash-4.1$ bart MANAGE -s ppas94 -c keep -i 1428768344061
INFO:  changing status of backup '1428768344061' of server 'ppas94' from
'active' to 'keep'
INFO:  3 WAL file(s) changed

-bash-4.1$ bart SHOW-BACKUPS -s ppas94 -i 1428768344061 -t
SERVER NAME    : ppas94
BACKUP ID      : 1428768344061
BACKUP NAME    : none
BACKUP STATUS  : keep
BACKUP TIME    : 2015-04-11 12:05:46 EDT
BACKUP SIZE    : 5.72 MB
WAL(S) SIZE    : 48.00 MB
NO. OF WALS    : 3
FIRST WAL FILE : 000000010000000100000025
CREATION TIME  : 2015-04-11 12:05:46 EDT
LAST WAL FILE  : 000000010000000100000027
CREATION TIME  : 2015-04-12 14:08:23 EDT
```

The following example resets the specified backup to active status:

```
-bash-4.1$ bart MANAGE -s ppas94 -c nokeep -i 1428768344061
INFO:  changing status of backup '1428768344061' of server 'ppas94' from
'keep' to 'active'
INFO:  3 WAL file(s) changed

-bash-4.1$ bart SHOW-BACKUPS -s ppas94 -i 1428768344061 -t
SERVER NAME    : ppas94
BACKUP ID      : 1428768344061
BACKUP NAME    : none
BACKUP STATUS  : active
BACKUP TIME    : 2015-04-11 12:05:46 EDT
BACKUP SIZE    : 5.72 MB
WAL(S) SIZE    : 48.00 MB
NO. OF WALS    : 3
FIRST WAL FILE : 000000010000000100000025
CREATION TIME  : 2015-04-11 12:05:46 EDT
LAST WAL FILE  : 000000010000000100000027
CREATION TIME  : 2015-04-12 14:08:23 EDT
```

The following example uses the enabled `wal_compression` parameter in the BART configuration file as shown by the following:

```
[PPAS94]
host = 127.0.0.1
port = 5444
user = enterprisedb
archive_command = 'cp %p %a/%f'
retention_policy = 3 BACKUPS
backup-name = ppas94_%year-%month-%dayT%hour:%minute:%second
wal_compression = enabled
description = "PPAS 94 server"
```

When the `MANAGE` subcommand is invoked, the following message is displayed indicating that WAL file compression is performed:

```
-bash-4.1$ bart MANAGE -s ppas94
INFO:  3 WAL file(s) compressed
INFO:  processing server 'ppas94', backup '1429219568720'
```

The following shows the archived WAL files in compressed format:

```
$ pwd
/opt/backup/ppas94
$ ls -l archived_wals
total 96
-rw------- 1 enterprisedb enterprisedb   305 Apr 15 16:00
000000010000000010000004A.00000028.backup
-rw------- 1 enterprisedb enterprisedb   305 Apr 16 17:26
000000010000000010000004F.00000028.backup
-rw------- 1 enterprisedb enterprisedb 27111 Apr 16 17:26 000000010000000010000004F.gz
-rw------- 1 enterprisedb enterprisedb 27094 Apr 17 15:09 0000000100000000100000050.gz
-rw------- 1 enterprisedb enterprisedb 27091 Apr 17 15:14 0000000100000000100000051.gz
-rw------- 1 enterprisedb enterprisedb   305 Apr 17 15:14
000000010000000100000052.00000028.backup
```

5.4.7 RESTORE

The `RESTORE` subcommand restores the base backup and its archived WAL files for the designated database server to the specified directory location. If the appropriate `RESTORE` options are specified, a `recovery.conf` file is generated with the recovery configuration parameters to perform point-in-time recovery.

Note: Use the `--debug` option prior to the `RESTORE` subcommand to see the commands executed during the process.

```
bart RESTORE -s server_name -p restore_path
  [ -i { backup_id | backup_name } ]
  [ -r remote_user@remote_host_address ]
  [ -t timeline_id ]
  [ { -x target_xid | -g target_timestamp } ]
  [ -c ]
```

Review the information (especially in Section 24.3.4 "Recovering Using a Continuous Archive Backup") documented in the *PostgreSQL Core Documentation* available at:

http://www.postgresql.org/docs/9.5/static/continuous-archiving.html

This reference material provides detailed information about the underlying point-in-time recovery process and the meaning and usage of the restore options that are generated into the `recovery.conf` file by BART.

Note: Check to ensure that the host where the backup is to be restored contains enough disk space for the base backup and its archived WAL files. The `RESTORE` subcommand does not have the capability to detect if there is sufficient disk space available before restoring the backup files. Thus, it is possible that the `RESTORE` subcommand may result in an error while copying files if there is not enough disk space available.

The steps for performing a restore operation are the following:

Step 1: Stop the Postgres database server on which you will be performing the restore operation.

Step 2: Inspect the `pg_xlog` subdirectory of the data directory and be sure to save any WAL files that have not yet been archived to the BART backup catalog (*`backup_path`*`/`*`server_name`*`/archived_wals`).

If there are some files that have not been archived, save these to a temporary location. You will later need to copy these files to the restored `pg_xlog` subdirectory after completing the `RESTORE` subcommand and before restarting the database server.

Step 3: Decide if you will restore to the current data directory, or to a new directory.

If you are restoring to the current data directory, delete all files and subdirectories under the data directory. For example, for an initial Postgres database server installation, this directory is *`POSTGRES_INSTALL_HOME`*`/data`.

If you are restoring to a new directory, create the directory on which to restore the backed up database cluster. Make sure the data directory can be written to by the BART user account or by the user account specified by the `remote-host` configuration parameter, or by the `--remote-host` option of the `RESTORE` subcommand if these are to be used.

Step 4: Perform the same process for tablespaces as described in Step 3. The `tablespace_path` parameter in the BART configuration file must contain the tablespace directory paths to which the tablespace data files are to be restored. See Section 4.2.4 for more information.

Step 5: Identify the timeline ID you wish to use to perform the restore operation.

The available timeline IDs can be identified by the first non-zero digit of the WAL file names reading from left to right.

In the following example, 1 is the only timeline ID in all of the available WAL files.

```
[root@localhost archived_wals]# pwd
/opt/backup/ppas94/archived wals
[root@localhost archived_wals]# ls -l
total 114692
-rw------- 1 enterprisedb enterprisedb 16777216 Nov 17 12:25 00000001000000000000004C
-rw------- 1 enterprisedb enterprisedb 16777216 Nov 17 12:25 00000001000000000000004D
-rw------- 1 enterprisedb enterprisedb 16777216 Nov 17 12:25 00000001000000000000004E
-rw------- 1 enterprisedb enterprisedb      305 Nov 17 12:25
00000001000000000000004E.00000028.backup
-rw------- 1 enterprisedb enterprisedb 16777216 Nov 17 12:25 00000001000000000000004F
-rw------- 1 enterprisedb enterprisedb 16777216 Nov 17 12:26 000000010000000000000050
-rw------- 1 enterprisedb enterprisedb 16777216 Nov 17 12:27 000000010000000000000051
-rw------- 1 enterprisedb enterprisedb 16777216 Nov 17 12:28 000000010000000000000052
```

Step 6: Identify the base backup to use for the restore operation and obtain the base backup ID. If you wish to use the latest (that is, the most recent) base backup, you can omit the `-i` option and the `RESTORE` subcommand uses that backup by default.

The base backups can be listed with the `SHOW-BACKUPS` subcommand as in the following example:

```
-bash-4.1$ bart SHOW-BACKUPS -s ppas94
SERVER NAME   BACKUP ID       BACKUP TIME              BACKUP SIZE   WAL(s) SIZE   WAL
FILES   STATUS

 ppas94        1447781103345   2015-11-17 12:25:03 EST   54.82 MB      80.00 MB      5
active
```

Step 7: Run the BART `RESTORE` subcommand.

If any of `-t` *`timeline_id`*, `-x` *`target_xid`*, or `-g` *`target_timestamp`* options are given, then a `recovery.conf` file is generated with the recovery configuration parameters corresponding to the specified options. That is, point-in-time recovery is performed upon restarting the database server.

If none of `-t` *`timeline_id`*, `-x` *`target_xid`*, and `-g` *`target_timestamp`* options are given, then no `recovery.conf` file is generated. In other words, only the base backup is restored and no point-in-time recovery is performed.

Use the `--debug` option to display the underlying commands used by BART.

Note: If invalid values are specified for the options, or if invalid option combinations are specified (for example, if both the `-x` *`target_xid`* and the `-g` *`target_timestamp`* options are given), no error message is generated by BART. The invalid options are accepted and passed to the `recovery.conf` file, which will then be processed by the database server when it is restarted.

Be sure that valid options are specified when using the RESTORE subcommand.

The following example uses the default, most recent backup by omitting the `-i` option:

```
-bash-4.1$ bart --debug RESTORE -s ppas94 -p /opt/restore
INFO:  restoring backup '1447781103345' of server 'ppas94'
DEBUG: Exec Command:  test -d /opt/restore && echo "exists"
DEBUG: Exec Command:  touch /opt/restore/tmp-1447781103345 && echo "exists"
DEBUG: Exec Command:  rm -f /opt/restore/tmp-1447781103345
DEBUG: Exec Command:  ls -A /opt/restore
INFO:  restoring backup to /opt/restore
DEBUG: restore command: cat /opt/backup/ppas94/1447781103345/base.tar |   tar
-C /opt/restore -xf -
DEBUG: Exec Command: cat /opt/backup/ppas94/1447781103345/base.tar |   tar -C
/opt/restore -xf -
INFO:  base backup restored
DEBUG: Backup Info file created at
'/opt/backup/ppas94/1447781103345/backupinfo'
DEBUG: Exec Command: echo "archive_mode = off" |   cat >>
/opt/restore/postgresql.conf
INFO:  archiving is disabled
```

Note: If the `-c` option is specified or if the enabled setting of the `copy_wals_during_restore` BART configuration parameter is in effect for this database server, then the following actions occur:

- In addition to restoring the database cluster to the directory specified by the `-p` *`restore_path`* option, the archived WAL files of the backup are copied from the BART backup catalog to the subdirectory *`restore_path`*`/archived_wals`.
- If a `recovery.conf` file is generated, the command string set in the `restore_command` parameter retrieves the WAL files from this `archived_wals` subdirectory relative to the *`restore_path`* parent directory as: `restore_command = 'cp archived_wals/%f %p'`

Step 8: Copy any saved WAL files from Step 2 to the *`restore_path`*`/pg_xlog` subdirectory.

Step 9: Inspect the restored directories and data files of the restored database cluster in directory *`restore_path`*.

All files and directories must be owned by the user account that you intend to use to start the database server. This user account is typically the Postgres user account (`enterprisedb` or `postgres`), but it may be some other user account of your choice. Recursively change the user and group ownership of the *`restore_path`* directory, its files, and its subdirectories if necessary.

There must only be directory access privileges for the user account that will start the database server. No other groups or users can have access to the directory.

Step 10: If one is generated, inspect the recovery configuration file named `recovery.conf` located in the *`restore_path`* directory to verify the parameter settings for a point-in-time recovery operation.

Step 11: WAL archiving is disabled at this point.

The BART `RESTORE` subcommand adds an `archive_mode = off` parameter at the end of the `postgresql.conf` file.

The following shows the end of the file where this parameter is added:

```
# Add settings for extensions here
archive_mode = off
```

If you want to restart the database server with WAL archiving activated, be sure to delete this additional parameter.

The original `archive_mode` parameter still resides in the `postgresql.conf` file in its initial location with its last setting.

Step 12: Start the database server to initiate recovery. After completion, check the database server log file to ensure the recovery was successful.

Options

`-s, --server` *`server_name`*

> *`server_name`* is the name of the database server to be restored.

`-p, --restore-path` *`restore_path`*

> *`restore_path`* is the directory path where the backup of the database server is to be restored. The directory must be empty and have the proper ownership and privileges assigned to it.

`-i, --backupid { backup_id | backup_name }`

> *`backup_id`* is the integer, base backup identifier of the base backup to be used for the restoration. *`backup_name`* is the user-defined alphanumeric name for the base backup. If the option is omitted, the default is to use the latest (that is, the most recent) base backup.

`-r, --remote-host` *`remote_user@remote_host_address`*

> *`remote_user`* is the user account on the remote database server host that accepts a password-less SSH/SCP login connection and is the owner of the directory

where the backup is to be restored. *remote_host_address* is the IP address of the remote host to which the backup is to be restored. This option must be specified if the remote-host parameter for this database server is not set in the BART configuration file.

-t, --target-tli *timeline_id*

timeline_id is the integer identifier of the timeline to be used for replaying the archived WAL files for point-in-time recovery.

-x, --target-xid *target_xid*

target_xid is the integer identifier of the transaction ID that determines the transaction up to and including, which point-in-time recovery encompasses. Only one of the -x *target_xid* or the -g *target_timestamp* option should be included if point-in-time recovery is desired.

-g, --target-timestamp *target_timestamp*

target_timestamp is the timestamp that determines the point in time up to and including, which point-in-time recovery encompasses. Only one of the -x *target_xid* or the -g *target_timestamp* option should be included if point-in-time recovery is desired.

-c, --copy-wals

If specified, the archived WAL files are copied from the BART backup catalog to directory *restore_path*/archived_wals. If BART generates the recovery.conf file for point-in-time recovery, the restore_command retrieves the WAL files from *restore_path*/archived_wals for the database server archive recovery. If the -c option is omitted and the copy_wals_during_restore parameter in the BART configuration file is not enabled in a manner applicable to this database server, the default action is that the restore_command in the recovery.conf file is generated to retrieve the archived WAL files directly from the BART backup catalog. See sections 4.1 and 4.2.5 for information on the copy_wals_during_restore parameter.

Example

The following example restores database server ppas93_remote to the /opt/restore directory up to timestamp 2015-12-15 10:47:00.

```
-bash-4.1$ bart RESTORE -s ppas93 remote -i 1450194208824 -p /opt/restore -t 1 -g
'2015-12-15 10:47:00'
INFO:  restoring backup '1450194208824' of server 'ppas93 remote'
INFO:  restoring backup to enterprisedb@192.168.2.24:/opt/restore
INFO:  base backup restored
```

```
INFO:  WAL file(s) will be streamed from the BART host
INFO:  creating recovery.conf file
INFO:  archiving is disabled
INFO:  tablespace(s) restored
```

The following is the content of the generated `recovery.conf` file:

```
restore_command = 'scp -o BatchMode=yes -o PasswordAuthentication=no
enterprisedb@192.168.2.22:/opt/backup/ppas93 remote/archived wals/%f %p'
recovery_target_time = '2015-12-15 10:47:00'
recovery_target_timeline = 1
```

The following displays the restored files and subdirectories.

```
[root@localhost restore]# pwd
/opt/restore
[root@localhost restore]# ls -l
total 108
-rw------- 1 enterprisedb enterprisedb   208 Dec 15 10:43 backup_label
drwx------ 6 enterprisedb enterprisedb  4096 Dec  2 10:38 base
drwx------ 2 enterprisedb enterprisedb  4096 Dec 15 10:42 dbms pipe
drwx------ 2 enterprisedb enterprisedb  4096 Dec 15 11:00 global
drwx------ 2 enterprisedb enterprisedb  4096 Nov 10 15:38 pg clog
-rw------- 1 enterprisedb enterprisedb  4438 Dec  2 10:38 pg_hba.conf
-rw------- 1 enterprisedb enterprisedb  1636 Nov 10 15:38 pg_ident.conf
drwxr-xr-x 2 enterprisedb enterprisedb  4096 Dec 15 10:42 pg_log
drwx------ 4 enterprisedb enterprisedb  4096 Nov 10 15:38 pg multixact
drwx------ 2 enterprisedb enterprisedb  4096 Dec 15 10:42 pg notify
drwx------ 2 enterprisedb enterprisedb  4096 Nov 10 15:38 pg_serial
drwx------ 2 enterprisedb enterprisedb  4096 Nov 10 15:38 pg_snapshots
drwx------ 2 enterprisedb enterprisedb  4096 Dec 15 10:42 pg_stat
drwx------ 2 enterprisedb enterprisedb  4096 Dec 15 10:43 pg_stat_tmp
drwx------ 2 enterprisedb enterprisedb  4096 Nov 10 15:38 pg_subtrans
drwx------ 2 enterprisedb enterprisedb  4096 Dec 15 11:00 pg tblspc
drwx------ 2 enterprisedb enterprisedb  4096 Nov 10 15:38 pg twophase
-rw------- 1 enterprisedb enterprisedb     4 Nov 10 15:38 PG_VERSION
drwx------ 2 enterprisedb enterprisedb  4096 Dec 15 11:00 pg_xlog
-rw------- 1 enterprisedb enterprisedb 23906 Dec 15 11:00 postgresql.conf
-rw-r--r-- 1 enterprisedb enterprisedb   217 Dec 15 11:00 recovery.conf
```

Example

The following example performs the `RESTORE` operation with the `-c` option to copy the archived WAL files to the local *`restore_path`*`/archived_wals` directory. The `-d` debug option displays the commands executed during the process.

```
-bash-4.1$ bart -d RESTORE -c -s ppas93 remote -i 1450194208824 -p /opt/restore -t 1 -g
'2015-12-15 10:47:00'
INFO:  restoring backup '1450194208824' of server 'ppas93_remote'
DEBUG: Exec Command: ssh -o BatchMode=yes -o PasswordAuthentication=no
enterprisedb@192.168.2.24 exit
DEBUG: Exec Command: ssh -o BatchMode=yes -o PasswordAuthentication=no
enterprisedb@192.168.2.24 test -d /opt/restore && echo "exists"
DEBUG: Exec Command: ssh -o BatchMode=yes -o PasswordAuthentication=no
enterprisedb@192.168.2.24 touch /opt/restore/tmp-1450194208824 && echo "exists"
DEBUG: Exec Command: ssh -o BatchMode=yes -o PasswordAuthentication=no
enterprisedb@192.168.2.24 rm -f /opt/restore/tmp-1450194208824
DEBUG: Exec Command: ssh -o BatchMode=yes -o PasswordAuthentication=no
enterprisedb@192.168.2.24 ls -A /opt/restore
DEBUG: Exec Command: ssh -o BatchMode=yes -o PasswordAuthentication=no
enterprisedb@192.168.2.24 test -d /opt/restore_tblspc_1 && echo "exists"
DEBUG: Exec Command: ssh -o BatchMode=yes -o PasswordAuthentication=no
enterprisedb@192.168.2.24 touch /opt/restore_tblspc_1/tmp-1450194208824 && echo
"exists"
```

```
DEBUG: Exec Command: ssh -o BatchMode=yes -o PasswordAuthentication=no
enterprisedb@192.168.2.24 rm -f /opt/restore tblspc 1/tmp-1450194208824
DEBUG: Exec Command: ssh -o BatchMode=yes -o PasswordAuthentication=no
enterprisedb@192.168.2.24 ls -A /opt/restore tblspc 1
DEBUG: Exec Command: ssh -o BatchMode=yes -o PasswordAuthentication=no
enterprisedb@192.168.2.24 test -d /opt/restore_tblspc_2 && echo "exists"
DEBUG: Exec Command: ssh -o BatchMode=yes -o PasswordAuthentication=no
enterprisedb@192.168.2.24 touch /opt/restore tblspc 2/tmp-1450194208824 && echo
"exists"
DEBUG: Exec Command: ssh -o BatchMode=yes -o PasswordAuthentication=no
enterprisedb@192.168.2.24 rm -f /opt/restore_tblspc_2/tmp-1450194208824
DEBUG: Exec Command: ssh -o BatchMode=yes -o PasswordAuthentication=no
enterprisedb@192.168.2.24 ls -A /opt/restore_tblspc_2
INFO:  restoring backup to enterprisedb@192.168.2.24:/opt/restore
DEBUG: restore command: cat /opt/backup/ppas93_remote/1450194208824/base.tar | ssh -o
BatchMode=yes -o PasswordAuthentication=no enterprisedb@192.168.2.24 " tar -C
/opt/restore -xf - "
DEBUG: Exec Command: cat /opt/backup/ppas93_remote/1450194208824/base.tar | ssh -o
BatchMode=yes -o PasswordAuthentication=no enterprisedb@192.168.2.24 " tar -C
/opt/restore -xf - "
INFO:  base backup restored
DEBUG: Exec Command: ssh -o BatchMode=yes -o PasswordAuthentication=no
enterprisedb@192.168.2.24 " gzip --version "
INFO:  copying WAL file(s) to enterprisedb@192.168.2.24:/opt/restore/archived_wals
DEBUG: Exec Command: ssh -o BatchMode=yes -o PasswordAuthentication=no
enterprisedb@192.168.2.24 " mkdir -p /opt/restore/archived_wals " && scp -o
BatchMode=yes -o PasswordAuthentication=no  -r
/opt/backup/ppas93_remote/archived_wals/000000010000000000000071
/opt/backup/ppas93_remote/archived_wals/000000010000000000000070
/opt/backup/ppas93_remote/archived_wals/00000001000000000000006F
/opt/backup/ppas93_remote/archived_wals/00000001000000000000006E
/opt/backup/ppas93_remote/archived_wals/00000001000000000000006D
/opt/backup/ppas93 remote/archived wals/00000001000000000000006C
/opt/backup/ppas93_remote/archived_wals/00000001000000000000006B
/opt/backup/ppas93_remote/archived_wals/00000001000000000000006A
/opt/backup/ppas93_remote/archived_wals/000000010000000000000069
enterprisedb@192.168.2.24:/opt/restore/archived_wals
INFO:  creating recovery.conf file
DEBUG: recovery restore command: 'cp archived wals/%f "%p"'
DEBUG: Exec Command: echo "restore_command = 'cp archived_wals/%f "%p"'
recovery_target_time = '2015-12-15 10:47:00'
recovery_target_timeline = 1
" | ssh -o BatchMode=yes -o PasswordAuthentication=no enterprisedb@192.168.2.24 " cat
>> /opt/restore/recovery.conf "
DEBUG: Backup Info file created at '/opt/backup/ppas93 remote/1450194208824/backupinfo'
DEBUG: Exec Command: echo "archive_mode = off" | ssh -o BatchMode=yes -o
PasswordAuthentication=no enterprisedb@192.168.2.24 " cat >>
/opt/restore/postgresql.conf "
INFO:  archiving is disabled
DEBUG: Exec Command: ssh -o BatchMode=yes -o PasswordAuthentication=no
enterprisedb@192.168.2.24 test -d /opt/restore_tblspc_1 && echo "exists"
DEBUG: Exec Command: ssh -o BatchMode=yes -o PasswordAuthentication=no
enterprisedb@192.168.2.24 touch /opt/restore_tblspc_1/tmp-1450194208824 && echo
"exists"
DEBUG: Exec Command: ssh -o BatchMode=yes -o PasswordAuthentication=no
enterprisedb@192.168.2.24 rm -f /opt/restore tblspc 1/tmp-1450194208824
DEBUG: Exec Command: ssh -o BatchMode=yes -o PasswordAuthentication=no
enterprisedb@192.168.2.24 ls -A /opt/restore_tblspc_1
DEBUG: Exec Command: ssh -o BatchMode=yes -o PasswordAuthentication=no
enterprisedb@192.168.2.24 " mkdir -p /opt/restore_tblspc_1 "
DEBUG: Exec Command: cat /opt/backup/ppas93_remote/1450194208824/17032.tar | ssh -o
BatchMode=yes -o PasswordAuthentication=no enterprisedb@192.168.2.24 " tar -C
/opt/restore_tblspc_1 -xf - "
DEBUG: Exec Command: ssh -o BatchMode=yes -o PasswordAuthentication=no
enterprisedb@192.168.2.24 " cd /opt/restore/pg_tblspc; rm -f 17032; ln -sf
/opt/restore_tblspc_1 17032 "
DEBUG: Exec Command: ssh -o BatchMode=yes -o PasswordAuthentication=no
enterprisedb@192.168.2.24 test -d /opt/restore tblspc 2 && echo "exists"
DEBUG: Exec Command: ssh -o BatchMode=yes -o PasswordAuthentication=no
enterprisedb@192.168.2.24 touch /opt/restore tblspc 2/tmp-1450194208824 && echo
"exists"
```

```
DEBUG: Exec Command: ssh -o BatchMode=yes -o PasswordAuthentication=no
enterprisedb@192.168.2.24 rm -f /opt/restore tblspc 2/tmp-1450194208824
DEBUG: Exec Command: ssh -o BatchMode=yes -o PasswordAuthentication=no
enterprisedb@192.168.2.24 ls -A /opt/restore tblspc 2
DEBUG: Exec Command: ssh -o BatchMode=yes -o PasswordAuthentication=no
enterprisedb@192.168.2.24 " mkdir -p /opt/restore_tblspc_2 "
DEBUG: Exec Command: cat /opt/backup/ppas93_remote/1450194208824/17033.tar | ssh -o
BatchMode=yes -o PasswordAuthentication=no enterprisedb@192.168.2.24 " tar -C
/opt/restore_tblspc_2 -xf - "
DEBUG: Exec Command: ssh -o BatchMode=yes -o PasswordAuthentication=no
enterprisedb@192.168.2.24 " cd /opt/restore/pg_tblspc; rm -f 17033; ln -sf
/opt/restore_tblspc_2 17033 "
INFO:  tablespace(s) restored
```

The following is the content of the generated `recovery.conf` file:

```
restore_command = 'cp archived_wals/%f %p'
recovery_target_time = '2015-12-15 10:47:00'
recovery_target_timeline = 1
```

The following displays the restored files and subdirectories.

```
[root@localhost restore]# pwd
/opt/restore
[root@localhost restore]# ls -l
total 112
drwxr-xr-x 2 enterprisedb enterprisedb  4096 Dec 15 11:24 archived_wals
-rw------- 1 enterprisedb enterprisedb   208 Dec 15 10:43 backup_label
drwx------ 6 enterprisedb enterprisedb  4096 Dec  2 10:38 base
drwx------ 2 enterprisedb enterprisedb  4096 Dec 15 10:42 dbms_pipe
drwx------ 2 enterprisedb enterprisedb  4096 Dec 15 11:24 global
drwx------ 2 enterprisedb enterprisedb  4096 Nov 10 15:38 pg_clog
-rw------- 1 enterprisedb enterprisedb  4438 Dec  2 10:38 pg_hba.conf
-rw------- 1 enterprisedb enterprisedb  1636 Nov 10 15:38 pg_ident.conf
drwxr-xr-x 2 enterprisedb enterprisedb  4096 Dec 15 10:42 pg_log
drwx------ 4 enterprisedb enterprisedb  4096 Nov 10 15:38 pg_multixact
drwx------ 2 enterprisedb enterprisedb  4096 Dec 15 10:42 pg_notify
drwx------ 2 enterprisedb enterprisedb  4096 Nov 10 15:38 pg_serial
drwx------ 2 enterprisedb enterprisedb  4096 Nov 10 15:38 pg_snapshots
drwx------ 2 enterprisedb enterprisedb  4096 Dec 15 10:42 pg_stat
drwx------ 2 enterprisedb enterprisedb  4096 Dec 15 10:43 pg_stat_tmp
drwx------ 2 enterprisedb enterprisedb  4096 Nov 10 15:38 pg_subtrans
drwx------ 2 enterprisedb enterprisedb  4096 Dec 15 11:24 pg_tblspc
drwx------ 2 enterprisedb enterprisedb  4096 Nov 10 15:38 pg_twophase
-rw------- 1 enterprisedb enterprisedb     4 Nov 10 15:38 PG VERSION
drwx------ 2 enterprisedb enterprisedb  4096 Dec 15 11:24 pg_xlog
-rw------- 1 enterprisedb enterprisedb 23906 Dec 15 11:24 postgresql.conf
-rw-r--r-- 1 enterprisedb enterprisedb   118 Dec 15 11:24 recovery.conf
```

5.4.8 DELETE

The `DELETE` subcommand removes the subdirectory and data files from the BART backup catalog for the specified base backups along with its archived WAL files.

```
bart DELETE -s server_name
  -i { all |
       ']{ backup_id | backup_name },... }['
     }
  [ -n ]
```

Note that a specific database server must be specified.

Note: Do not invoke the `DELETE` subcommand while the BART `BACKUP` subcommand is in progress. Base backups affected by the backup process will be skipped and ignored by the `DELETE` subcommand.

For database servers under a retention policy, there are conditions where certain backups may not be deleted. See Section 5.2.4.1 for information regarding permitted backup deletions.

Options

`-s, --server` *`server_name`*

> *`server_name`* is the name of the database server whose backups are to be deleted.

`-i, --backupid { all | ['']{ backup_id | backup_name },... }['] }`

> *`backup_id`* is the integer, base backup identifier of the base backup to be deleted. *`backup_name`* is the user-defined alphanumeric name for the base backup. Multiple backup identifiers and backup names may be specified in a comma-separated list. The list must be enclosed within single quotes if there is any white space appearing before or after each comma. If `all` is specified, all of the base backups and their archived WAL files for the specified database server are deleted.

`-n, --dry-run`

> Displays the results as if the deletions were done, however, no physical removal of the files are actually performed. In other words, a test run is performed so that you can see the potential results prior to actually initiating the action.

Example

The following example deletes a backup from the specified database server.

```
$ bart DELETE -s ppas94 -i ppas94_2015-04-15T16:00:24
INFO:  deleting backup 'ppas94_2015-04-15T16:00:24' of server 'ppas94'
INFO:  deleting backup '1429128024311'
INFO:  4 WAL file(s) will be removed
INFO:  1 Unused WAL file(s) will be removed
INFO:  deleting WAL file '00000001000000010000004E'
INFO:  deleting WAL file '00000001000000010000004D'
INFO:  deleting WAL file '00000001000000010000004B'
INFO:  deleting WAL file '00000001000000010000004A'
INFO:  deleting (unused) WAL file '000000010000000100000042.00000028'
INFO:  backup(s) deleted
```

After the deletion, the BART backup catalog for the database server no longer contains the corresponding directory for the deleted base backup ID. The `archived_wals` subdirectory no longer contains the WAL files of the backup.

```
$ pwd
/opt/backup/ppas94
$ ls -l
total 8
drwx------ 2 enterprisedb enterprisedb 4096 Apr 20 10:09 1429219568720
drwx------ 2 enterprisedb enterprisedb 4096 Apr 20 10:21 archived_wals
$ ls -l archived wals
total 49164
-rw------- 1 enterprisedb enterprisedb      305 Apr 15 16:00
000000010000000010000004A.00000028.backup
-rw------- 1 enterprisedb enterprisedb 16777216 Apr 16 17:26 000000010000000010000004F
-rw------- 1 enterprisedb enterprisedb      305 Apr 16 17:26
000000010000000010000004F.00000028.backup
-rw------- 1 enterprisedb enterprisedb 16777216 Apr 17 15:09 000000010000000100000050
-rw------- 1 enterprisedb enterprisedb 16777216 Apr 17 15:14 000000010000000100000051
-rw------- 1 enterprisedb enterprisedb      305 Apr 17 15:14
000000010000000100000052.00000028.backup
```

The following example deletes multiple backups from the database server.

```
$ bart DELETE -s ppas94 -i 1428355371389,1428422324880,ppas94_2015-04-17T15:14:33
INFO:  deleting backup '1428355371389' of server 'ppas94'
INFO:  deleting backup '1428355371389'
INFO:  1 WAL file(s) will be removed
INFO:  deleting WAL file '0000000100000000000000AA'
INFO:  backup(s) deleted
INFO:  deleting backup '1428422324880' of server 'ppas94'
INFO:  deleting backup '1428422324880'
INFO:  2 WAL file(s) will be removed
INFO:  1 Unused WAL file(s) will be removed
INFO:  deleting WAL file '0000000100000000000000E1'
INFO:  deleting WAL file '0000000100000000000000E0'
INFO:  deleting (unused) WAL file '0000000100000000000000AA.00000028'
INFO:  backup(s) deleted
INFO:  deleting backup 'ppas94_2015-04-17T15:14:33' of server 'ppas94'
INFO:  deleting backup '1429298073247'
INFO:  1 WAL file(s) will be removed
INFO:  deleting WAL file '000000010000000100000052'
INFO:  backup(s) deleted
```

The following example also deletes multiple backups, but since there are space characters in the comma-separated list, the entire list must be enclosed within single quotes.

```
$ bart DELETE -s ppas94 -i '1428502049836, 1428589759899, 1428684537299'
INFO:  deleting backup '1428502049836' of server 'ppas94'
INFO:  deleting backup '1428502049836'
INFO:  6 WAL file(s) will be removed
INFO:  deleting WAL file '000000010000000100000003'
INFO:  deleting WAL file '000000010000000100000002'
INFO:  deleting WAL file '000000010000000100000001'
INFO:  deleting WAL file '000000010000000100000000'
INFO:  deleting WAL file '0000000100000000000000E3'
INFO:  deleting WAL file '0000000100000000000000E2'
INFO:  backup(s) deleted
INFO:  deleting backup '1428589759899' of server 'ppas94'
INFO:  deleting backup '1428589759899'
INFO:  6 WAL file(s) will be removed
INFO:  1 Unused WAL file(s) will be removed
INFO:  deleting WAL file '000000010000000100000009'
INFO:  deleting WAL file '000000010000000100000008'
INFO:  deleting WAL file '000000010000000100000007'
```

```
INFO:  deleting WAL file '000000010000000100000006'
INFO:  deleting WAL file '000000010000000100000005'
INFO:  deleting WAL file '000000010000000100000004'
INFO:  deleting (unused) WAL file '000000010000000000000E2.00000028'
INFO:  backup(s) deleted
INFO:  deleting backup '1428684537299' of server 'ppas94'
INFO:  deleting backup '1428684537299'
INFO:  17 WAL file(s) will be removed
INFO:  2 Unused WAL file(s) will be removed
INFO:  deleting WAL file '000000010000000100000024'
INFO:  deleting WAL file '000000010000000100000023'
INFO:  deleting WAL file '000000010000000100000022'
INFO:  deleting WAL file '000000010000000100000021'
INFO:  deleting WAL file '000000010000000100000020'
INFO:  deleting WAL file '00000001000000010000001F'
INFO:  deleting WAL file '00000001000000010000001E'
INFO:  deleting WAL file '00000001000000010000001D'
INFO:  deleting WAL file '00000001000000010000001C'
INFO:  deleting WAL file '00000001000000010000001B'
INFO:  deleting WAL file '00000001000000010000001A'
INFO:  deleting WAL file '000000010000000100000019'
INFO:  deleting WAL file '000000010000000100000018'
INFO:  deleting WAL file '000000010000000100000017'
INFO:  deleting WAL file '000000010000000100000016'
INFO:  deleting WAL file '000000010000000100000015'
INFO:  deleting WAL file '000000010000000100000014'
INFO:  deleting (unused) WAL file '000000010000000100000004.00000028'
INFO:  deleting (unused) WAL file '00000001000000010000000A.00000028'
INFO:  backup(s) deleted
```

6 Sample BART System with Local and Remote Database Servers

This chapter describes a sample BART managed backup and recovery system consisting of both local and remote database servers. The complete steps to configure and operate the system are provided.

For explanations of the configuration steps, refer to Chapter 4. For information about the operational procedures and BART subcommands, see Chapter 5.

The environment for this sample system is as follows:

- BART on host 192.168.2.22 running with BART user account `enterprisedb`
- Local Advanced Server on host 192.168.2.22 running with user account `enterprisedb`
- Remote Advanced Server on host 192.168.2.24 running with user account `enterprisedb`
- Remote PostgreSQL server on host 192.168.2.24 running with user account `postgres`

Password-less SSH/SCP connections are required between the following:

- BART on host 192.168.2.22 and the remote Advanced Server on host 192.168.2.24
- BART on host 192.168.2.22 and the remote PostgreSQL server on host 192.168.2.24

Note that SSH/SCP is not used between the BART host and the local Advanced Server for the following reasons:

- The BART host and the local Advanced Server are both on host 192.168.2.22, which is also where the intended restore path is located for the local Advanced Server database cluster, and
- The BART user account (and thus, the owner of the BART backup catalog directory) is `enterprisedb`, and the user account running Advanced Server is `enterprisedb` as well.

The following sections show the configuration steps and operation for this system.

6.1 BART Configuration File

The following shows the settings used in the BART configuration file:

```
[BART]
```



```
bart-host= enterprisedb@192.168.2.22
backup_path = /opt/backup
pg_basebackup_path = /opt/PostgresPlus/9.4AS/bin/pg_basebackup
retention_policy = 6 BACKUPS
logfile = /tmp/bart.log

[PPAS94]
host = 127.0.0.1
port = 5444
user = enterprisedb
backup-name = ppas94_%year-%month-%dayT%hour:%minute
archive_command = 'cp %p %a/%f'
description = "PPAS 94 server"

[PPAS93_remote]
host = 192.168.2.24
port = 5444
user = repuser
backup-name = ppas93_%year-%month-%dayT%hour:%minute
remote-host = enterprisedb@192.168.2.24
description = "PPAS 93 remote server"

[PG94_remote]
host = 192.168.2.24
port = 5432
user = postgres
backup-name = pg94_%year-%month-%dayT%hour:%minute
remote-host = postgres@192.168.2.24
copy_wals_during_restore = enabled
description = "PostgreSQL 94 remote server"
```

6.2 SSH/SCP Password-Less Connections

This section shows how the password-less SSH/SCP connections were established with the authorized public keys files.

6.2.1 Generation of Public Key File for the BART User Account

The BART user account is `enterprisedb` with the home directory of `/opt/PostgresPlus/9.4AS`.

Generation of the public key file is as follows. First, create the `.ssh` subdirectory in the BART user's home directory:

```
[root@localhost 9.4AS]# pwd
/opt/PostgresPlus/9.4AS
[root@localhost 9.4AS]# mkdir .ssh
[root@localhost 9.4AS]# chown enterprisedb .ssh
[root@localhost 9.4AS]# chgrp enterprisedb .ssh
[root@localhost 9.4AS]# chmod 700 .ssh
[root@localhost 9.4AS]# ls -la | grep ssh
drwx------  2 enterprisedb enterprisedb    4096 Apr 23 13:02 .ssh
```

Make sure there are no groups or other users that can access the `.ssh` directory.

Generate the public key file.

```
[user@localhost ~]$ su - enterprisedb
Password:
-bash-4.1$ pwd
/opt/PostgresPlus/9.4AS
-bash-4.1$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/opt/PostgresPlus/9.4AS/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /opt/PostgresPlus/9.4AS/.ssh/id_rsa.
Your public key has been saved in /opt/PostgresPlus/9.4AS/.ssh/id_rsa.pub.
The key fingerprint is:
de:65:34:d6:b1:d2:32:3c:b0:43:c6:a3:c0:9f:f4:64
enterprisedb@localhost.localdomain
The key's randomart image is:
+--[ RSA 2048]----+
|     .   .+   .  |
|      o .oE+ o o |
|       + *o.X +  |
|        + .+ *   |
|        S   o    |
|       . . o     |
|        . .      |
|                 |
|                 |
+-----------------+
```

The following are the resulting files. `id_rsa.pub` is the public key file of BART user account `enterprisedb`.

```
-bash-4.1$ ls -l .ssh
total 8
-rw------- 1 enterprisedb enterprisedb 1675 Apr 23 13:04 id_rsa
-rw-r--r-- 1 enterprisedb enterprisedb  416 Apr 23 13:04 id_rsa.pub
```

6.2.2 Set Up Access from Remote Advanced Server to BART Host

On the remote host 192.168.2.24, create the public key file for the remote database server user account, `enterprisedb`, for access to the BART user account, `enterprisedb`, on the BART host 192.168.2.22.

This is for scenario 1 as described in Section 4.2.1.3.

Create the `.ssh` directory for user account `enterprisedb` on the remote host:

```
[root@localhost 9.3AS]# pwd
/opt/PostgresPlus/9.3AS
[root@localhost 9.3AS]# mkdir .ssh
[root@localhost 9.3AS]# chown enterprisedb .ssh
[root@localhost 9.3AS]# chgrp enterprisedb .ssh
[root@localhost 9.3AS]# chmod 700 .ssh
[root@localhost 9.3AS]# ls -la | grep ssh
```

```
drwx------  2 enterprisedb enterprisedb    4096 Apr 23 13:08 .ssh
```

Generate the public key file on the remote host for user account `enterprisedb`.

```
[user@localhost ~]$ su - enterprisedb
Password:
-bash-4.1$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/opt/PostgresPlus/9.3AS/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /opt/PostgresPlus/9.3AS/.ssh/id_rsa.
Your public key has been saved in /opt/PostgresPlus/9.3AS/.ssh/id_rsa.pub.
The key fingerprint is:
15:27:1e:1e:61:4b:48:66:67:0b:b2:be:fc:ea:ea:e6
enterprisedb@localhost.localdomain
The key's randomart image is:
+--[ RSA 2048]----+
|      ..=.@..    |
|       =.O O     |
|      .   *      |
|     .   .       |
|      . S        |
|     . .         |
|      o          |
|    .  .         |
|   +Eoo..        |
+-----------------+
```

Copy the generated public key file, `id_rsa.pub`, to the BART user account, `enterprisedb`, on the BART host, `192.168.2.22`:

```
-bash-4.1$ scp ~/.ssh/id_rsa.pub enterprisedb@192.168.2.22:/tmp/tmp.pub
The authenticity of host '192.168.2.22 (192.168.2.22)' can't be established.
RSA key fingerprint is b8:a9:97:31:79:16:b8:2b:b0:60:5a:91:38:d7:68:22.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.2.22' (RSA) to the list of known hosts.
enterprisedb@192.168.2.22's password:
id_rsa.pub
```

Log into the BART host as the BART user account and append the temporary public key file, `/tmp/tmp.pub` onto the `authorized_keys` file owned by the BART user account.

```
-bash-4.1$ ssh enterprisedb@192.168.2.22
enterprisedb@192.168.2.22's password:
Last login: Tue Apr 21 17:03:24 2015 from 192.168.2.22
-bash-4.1$ pwd
/opt/PostgresPlus/9.4AS
-bash-4.1$ cat /tmp/tmp.pub >> ~/.ssh/authorized_keys
-bash-4.1$ ls -l .ssh
total 12
-rw-rw-r-- 1 enterprisedb enterprisedb  416 Apr 23 13:15 authorized_keys
-rw------- 1 enterprisedb enterprisedb 1675 Apr 23 13:04 id_rsa
-rw-r--r-- 1 enterprisedb enterprisedb  416 Apr 23 13:04 id_rsa.pub
```

The `authorized_keys` file must have file permission 600 as set by the following `chmod 600` command, otherwise the password-less connection fails:

```
-bash-4.1$ chmod 600 ~/.ssh/authorized_keys
-bash-4.1$ ls -l .ssh
total 12
-rw------- 1 enterprisedb enterprisedb  416 Apr 23 13:15 authorized_keys
-rw------- 1 enterprisedb enterprisedb 1675 Apr 23 13:04 id_rsa
-rw-r--r-- 1 enterprisedb enterprisedb  416 Apr 23 13:04 id_rsa.pub
-bash-4.1$ rm /tmp/tmp.pub
-bash-4.1$ exit
logout
Connection to 192.168.2.22 closed.
```

Test the password-less connection. From the remote host, verify that you can log into the BART host with the BART user account without being prompted for a password:

```
-bash-4.1$ ssh enterprisedb@192.168.2.22
Last login: Thu Apr 23 13:14:48 2015 from 192.168.2.24
-bash-4.1$ exit
logout
Connection to 192.168.2.22 closed.
```

6.2.3 Set Up Access from BART Host to Remote Advanced Server

On the BART host 192.168.2.22, copy the public key file for the BART user account, `enterprisedb`, for access to the remote database server user account, `enterprisedb`, on the remote host 192.168.2.24.

This is for scenario 2 as described in Section 4.2.1.3.

The following lists the current SSH keys files in the BART user's `.ssh` directory on the BART host:

```
[user@localhost ~]$ su - enterprisedb
Password:
-bash-4.1$ pwd
/opt/PostgresPlus/9.4AS
-bash-4.1$ ls -l .ssh
total 12
-rw------- 1 enterprisedb enterprisedb  416 Apr 23 13:15 authorized_keys
-rw------- 1 enterprisedb enterprisedb 1675 Apr 23 13:04 id_rsa
-rw-r--r-- 1 enterprisedb enterprisedb  416 Apr 23 13:04 id_rsa.pub
```

The public key file, `id_rsa.pub`, for BART user account `enterprisedb` on the BART host was generated in Section 6.2.1, and is now copied to the remote Advanced Server host on 192.168.2.24:

```
-bash-4.1$ scp ~/.ssh/id_rsa.pub enterprisedb@192.168.2.24:/tmp/tmp.pub
The authenticity of host '192.168.2.24 (192.168.2.24)' can't be established.
RSA key fingerprint is 59:41:fb:0c:ae:64:3d:3f:a2:d9:90:95:cf:2c:99:f2.
```

```
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.2.24' (RSA) to the list of known hosts.
enterprisedb@192.168.2.24's password:
id_rsa.pub
```

Log into the `enterprisedb` user account on the remote host and copy the public key file onto the `authorized_keys` file of the remote `enterprisedb` user account under its `.ssh` directory:

```
-bash-4.1$ ssh enterprisedb@192.168.2.24
enterprisedb@192.168.2.24's password:
Last login: Tue Apr 21 09:53:18 2015 from 192.168.2.22
-bash-4.1$ pwd
/opt/PostgresPlus/9.3AS
-bash-4.1$ ls -l .ssh
total 12
-rw------- 1 enterprisedb enterprisedb 1675 Apr 23 13:11 id_rsa
-rw-r--r-- 1 enterprisedb enterprisedb  416 Apr 23 13:11 id_rsa.pub
-rw-r--r-- 1 enterprisedb enterprisedb  394 Apr 23 13:12 known_hosts
-bash-4.1$ cat /tmp/tmp.pub >> ~/.ssh/authorized_keys
```

Adjust the file permission on `authorized_keys`.

```
-bash-4.1$ chmod 600 ~/.ssh/authorized_keys
-bash-4.1$ ls -l .ssh
total 16
-rw------- 1 enterprisedb enterprisedb  416 Apr 23 13:26 authorized_keys
-rw------- 1 enterprisedb enterprisedb 1675 Apr 23 13:11 id_rsa
-rw-r--r-- 1 enterprisedb enterprisedb  416 Apr 23 13:11 id_rsa.pub
-rw-r--r-- 1 enterprisedb enterprisedb  394 Apr 23 13:12 known_hosts
-bash-4.1$ rm /tmp/tmp.pub
-bash-4.1$ exit
logout
Connection to 192.168.2.24 closed.
```

While logged into the BART host, test the password-less connection from the BART host to the remote Advanced Server host.

```
-bash-4.1$ ssh enterprisedb@192.168.2.24
Last login: Thu Apr 23 13:25:53 2015 from 192.168.2.22
-bash-4.1$ exit
logout
Connection to 192.168.2.24 closed.
```

6.2.4 Set Up Access from Remote PostgreSQL to BART Host

On the remote host 192.168.2.24, create the public key file for the remote database server user account, `postgres`, for access to the BART user account, `enterprisedb`, on the BART host 192.168.2.22.

This is for scenario 1 as described in Section 4.2.1.3.

Create the `.ssh` directory for user account `postgres` on the remote host:

```
[root@localhost 9.4]# cd /opt/PostgreSQL/9.4
[root@localhost 9.4]# mkdir .ssh
[root@localhost 9.4]# chown postgres .ssh
[root@localhost 9.4]# chgrp postgres .ssh
[root@localhost 9.4]# chmod 700 .ssh
[root@localhost 9.4]# ls -la | grep ssh
drwx------  2 postgres postgres    4096 Apr 23 13:32 .ssh
```

Create and copy the generated public key file, `id_rsa.pub`, to the BART user account, `enterprisedb`, on the BART host, `192.168.2.22`:

```
[user@localhost ~]$ su - postgres
Password:
-bash-4.1$ pwd
/opt/PostgreSQL/9.4

-bash-4.1$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/opt/PostgreSQL/9.4/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /opt/PostgreSQL/9.4/.ssh/id_rsa.
Your public key has been saved in /opt/PostgreSQL/9.4/.ssh/id_rsa.pub.
The key fingerprint is:
1f:f8:76:d6:fc:a5:1a:c5:5a:66:66:01:d0:a0:ca:ba
postgres@localhost.localdomain
The key's randomart image is:
+--[ RSA 2048]----+
|          o+.    |
|         .  ..   |
|        .     .  |
|     . . .   . . |
|      o S .   O  |
|     .   o . @   |
|    .     + = o .|
|     .   . o . o.|
|    E       ... .|
+-----------------+

-bash-4.1$ ls -l .ssh
total 8
-rw------- 1 postgres postgres 1671 Apr 23 13:36 id_rsa
-rw-r--r-- 1 postgres postgres  412 Apr 23 13:36 id_rsa.pub

-bash-4.1$ scp ~/.ssh/id_rsa.pub enterprisedb@192.168.2.22:/tmp/tmp.pub
The authenticity of host '192.168.2.22 (192.168.2.22)' can't be established.
RSA key fingerprint is b8:a9:97:31:79:16:b8:2b:b0:60:5a:91:38:d7:68:22.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.2.22' (RSA) to the list of known hosts.
enterprisedb@192.168.2.22's password:
id_rsa.pub
```

Log into the BART host as the BART user account and append the temporary public key file, `/tmp/tmp.pub`, onto the `authorized_keys` file owned by the BART user account.

```
-bash-4.1$ ssh enterprisedb@192.168.2.22
```

```
enterprisedb@192.168.2.22's password:
Last login: Thu Apr 23 13:19:25 2015 from 192.168.2.24
-bash-4.1$ pwd
/opt/PostgresPlus/9.4AS
-bash-4.1$ cat /tmp/tmp.pub >> ~/.ssh/authorized_keys
-bash-4.1$ ls -l .ssh
total 16
-rw------- 1 enterprisedb enterprisedb  828 Apr 23 13:40 authorized_keys
-rw------- 1 enterprisedb enterprisedb 1675 Apr 23 13:04 id_rsa
-rw-r--r-- 1 enterprisedb enterprisedb  416 Apr 23 13:04 id_rsa.pub
-rw-r--r-- 1 enterprisedb enterprisedb  394 Apr 23 13:24 known_hosts
-bash-4.1$ rm /tmp/tmp.pub
-bash-4.1$ exit
logout
Connection to 192.168.2.22 closed.
```

Make sure the `authorized_keys` file has file permission 600 as shown, otherwise the password-less connection fails.

Test the password-less connection. From the remote host while logged in as user account `postgres`, verify that you can log into the BART host with the BART user account without being prompted for a password:

```
-bash-4.1$ pwd
/opt/PostgreSQL/9.4
-bash-4.1$ ssh enterprisedb@192.168.2.22
Last login: Thu Apr 23 13:40:10 2015 from 192.168.2.24
-bash-4.1$ exit
logout
Connection to 192.168.2.22 closed.
```

6.2.5 Set Up Access from BART Host to Remote PostgreSQL

On the BART host 192.168.2.22, copy the public key file for the BART user account, `enterprisedb`, for access to the remote database server user account, `postgres`, on the remote host 192.168.2.24.

This is for scenario 2 as described in Section 4.2.1.3.

The following lists the current SSH keys files in the BART user's `.ssh` directory on the BART host:

```
[user@localhost ~]$ su - enterprisedb
Password:
-bash-4.1$ ls -l .ssh
total 16
-rw------- 1 enterprisedb enterprisedb  828 Apr 23 13:40 authorized_keys
-rw------- 1 enterprisedb enterprisedb 1675 Apr 23 13:04 id_rsa
-rw-r--r-- 1 enterprisedb enterprisedb  416 Apr 23 13:04 id_rsa.pub
-rw-r--r-- 1 enterprisedb enterprisedb  394 Apr 23 13:24 known_hosts
```

The public key file, `id_rsa.pub`, for BART user account `enterprisedb` on the BART host was generated in Section 6.2.1, and is now copied to the remote PostgreSQL host on 192.168.2.24:

```
-bash-4.1$ scp ~/.ssh/id_rsa.pub postgres@192.168.2.24:/tmp/tmp.pub
postgres@192.168.2.24's password:
id_rsa.pub
```

Log into the `postgres` user account on the remote host and copy the public key file onto the `authorized_keys` file of `postgres` under its `.ssh` directory:

```
-bash-4.1$ ssh postgres@192.168.2.24
postgres@192.168.2.24's password:
Last login: Mon Jan 26 18:08:36 2015 from 192.168.2.19
-bash-4.1$ pwd
/opt/PostgreSQL/9.4
-bash-4.1$ cat /tmp/tmp.pub >> ~/.ssh/authorized_keys
```

Adjust the file permissions on `authorized_keys`.

```
-bash-4.1$ ls -l .ssh
total 16
-rw-rw-r-- 1 postgres postgres  416 Apr 23 13:52 authorized_keys
-rw------- 1 postgres postgres 1671 Apr 23 13:36 id_rsa
-rw-r--r-- 1 postgres postgres  412 Apr 23 13:36 id_rsa.pub
-rw-r--r-- 1 postgres postgres  394 Apr 23 13:36 known_hosts
-bash-4.1$ chmod 600 ~/.ssh/authorized_keys
-bash-4.1$ ls -l .ssh
total 16
-rw------- 1 postgres postgres  416 Apr 23 13:52 authorized_keys
-rw------- 1 postgres postgres 1671 Apr 23 13:36 id_rsa
-rw-r--r-- 1 postgres postgres  412 Apr 23 13:36 id_rsa.pub
-rw-r--r-- 1 postgres postgres  394 Apr 23 13:36 known_hosts
-bash-4.1$ rm /tmp/tmp.pub
-bash-4.1$ exit
logout
Connection to 192.168.2.24 closed.
```

Test the password-less connection from the BART host to the remote PostgreSQL host.

```
[user@localhost ~]$ su - enterprisedb
Password:
-bash-4.1$ ssh postgres@192.168.2.24
Last login: Thu Apr 23 13:52:25 2015 from 192.168.2.22
-bash-4.1$ exit
logout
Connection to 192.168.2.24 closed.
```

6.3 Replication Database User

This section shows how the replication database user is established for usage by the `pg_basebackup` program.

The replication database user for each database server is specified by the `user` parameter in the BART configuration file as shown by the following:

```
[PPAS94]
host = 127.0.0.1
port = 5444
user = enterprisedb        <=== Replication Database User
backup-name = ppas94_%year-%month-%dayT%hour:%minute
archive_command = 'cp %p %a/%f'
description = "PPAS 94 server"

[PPAS93_remote]
host = 192.168.2.24
port = 5444
user = repuser             <=== Replication Database User
backup-name = ppas93_%year-%month-%dayT%hour:%minute
remote-host = enterprisedb@192.168.2.24
description = "PPAS 93 remote server"

[PG94_remote]
host = 192.168.2.24
port = 5432
user = postgres            <=== Replication Database User
backup-name = pg94_%year-%month-%dayT%hour:%minute
remote-host = postgres@192.168.2.24
copy_wals_during_restore = enabled
description = "PostgreSQL 94 remote server"
```

To allow the required no prompt for password access to each database server when the BART user account initiates `pg_basebackup`, the `.pgpass` file contains the following entries. This file is for BART user account, `enterprisedb`, located in its home directory, `/opt/PostgresPlus/9.4AS/.pgpass`.

```
127.0.0.1:5444:*:enterprisedb:password
192.168.2.24:5444:*:repuser:password
192.168.2.24:5432:*:postgres:password
```

For the remote Advanced Server, `PPAS93_remote`, the replication database user is a specially created role with the `replication` privilege. All other database servers use a superuser as the replication database user.

While connected to `PPAS93_remote` on 192.168.2.24, the following `CREATE ROLE` command is given to create the replication database user:

```
CREATE ROLE repuser WITH LOGIN REPLICATION PASSWORD 'password';
```

This database user also has the `CONNECT` privilege on all databases.

The `pg_hba.conf` file for the local Advanced Server, `PPAS94` is set as follows:

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
local   all             all                                     md5
```

```
# IPv4 local connections:
host    template1       enterprisedb    127.0.0.1/32           md5
host    edb             enterprisedb    127.0.0.1/32           md5
#host   all             all             127.0.0.1/32           md5
# IPv6 local connections:
host    all             all             ::1/128                md5
# Allow replication connections from localhost, by a user with the
# replication privilege.
#local   replication    enterprisedb                           md5
host     replication    enterprisedb    127.0.0.1/32           md5
```

The `pg_hba.conf` file for the remote Advanced Server, `PPAS93_remote` is set as follows:

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
local   all             all                                     md5
# IPv4 local connections:
host    template1       repuser         192.168.2.22/32         md5
host    all             enterprisedb    127.0.0.1/32            md5
#host   all             all             127.0.0.1/32            md5
# IPv6 local connections:
host    all             all             ::1/128                 md5
# Allow replication connections from localhost, by a user with the
# replication privilege.
#local   replication    enterprisedb                            md5
host     replication    repuser         192.168.2.22/32         md5
```

The `pg_hba.conf` file for the remote PostgreSQL server, `PG94_remote` is set as follows:

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
local   all             all                                     md5
# IPv4 local connections:
host    template1       postgres        192.168.2.22/32         md5
host    all             all             127.0.0.1/32            md5
# IPv6 local connections:
host    all             all             ::1/128                 md5
# Allow replication connections from localhost, by a user with the
# replication privilege.
#local   replication    postgres                                md5
host     replication    postgres        192.168.2.22/32         md5
```

6.4 WAL Archiving Configuration Parameters

The parameters in the `postgresql.conf` file to enable WAL archiving are shown by the following.

The `postgresql.conf` file for the local Advanced Server, `PPAS94` is set as follows:

```
wal_level = archive
```

```
archive_mode = on               # allows archiving to be done
                                # (change requires restart)
#archive_command = ''          # command to use to archive a logfile segment
                                # placeholders: %p = path of file to archive
                                #               %f = file name only

archive_timeout = 50
max_wal_senders = 3
```

Since this database server is version 9.4, when the `INIT` subcommand is invoked, the Postgres `archive_command` configuration parameter in the `postgresql.auto.conf` file will be set based on the BART `archive_command` parameter located in the BART configuration file:

```
[BART]
bart-host= enterprisedb@192.168.2.22
backup_path = /opt/backup
pg_basebackup_path = /opt/PostgresPlus/9.4AS/bin/pg_basebackup
retention_policy = 6 BACKUPS
logfile = /tmp/bart.log

[PPAS94]
host = 127.0.0.1
port = 5444
user = enterprisedb
backup-name = ppas94_%year-%month-%dayT%hour:%minute
archive_command = 'cp %p %a/%f'
description = "PPAS 94 server"
```

The `postgresql.auto.conf` file contains the following after the `INIT` subcommand is invoked:

```
# Do not edit this file manually!
# It will be overwritten by ALTER SYSTEM command.
archive_command = 'cp %p /opt/backup/ppas94/archived_wals/%f'
```

The `archive_command` uses the `cp` command instead of `scp` since the BART backup catalog is local to this database cluster and the BART user account owning the backup catalog, `enterprisedb`, is the same user account running Advanced Server. The result is that there is no directory permission conflict during the archive operation.

Note: `archive_timeout` is set to 50 seconds to create more WAL files for illustrative purposes of this sample system.

The `postgresql.conf` file for the remote Advanced Server, `PPAS93_remote` is set as follows:

```
wal_level = archive
archive_mode = on               # allows archiving to be done
                                # (change requires restart)
archive_command =
'scp %p enterprisedb@192.168.2.22:/opt/backup/ppas93_remote/archived_wals/%f'
                                # placeholders: %p = path of file to archive
                                #               %f = file name only
```

```
archive_timeout = 50
max_wal_senders = 3
```

Since this database server is prior to version 9.4, the Postgres `archive_command` parameter must be manually set in the `postgresql.conf` file.

The `archive_command` uses the `scp` command since the BART backup catalog is remote relative to this database cluster. The BART user account, `enterprisedb`, is specified on the `scp` command since this is the user account owning the BART backup catalog where the archived WAL files are to be copied. The result is that there is no directory permission conflict during the archive operation.

The `postgresql.conf` file for the remote PostgreSQL server, `PG94_remote` is set as follows:

```
wal_level = archive
archive_mode = on               # allows archiving to be done
                                # (change requires restart)
#archive_command = ''           # command to use to archive a logfile segment
                                # placeholders: %p = path of file to archive
                                #               %f = file name only

archive_timeout = 50
max_wal_senders = 3
```

Since this database server is version 9.4, when the `INIT` subcommand is invoked, the Postgres `archive_command` configuration parameter in the `postgresql.auto.conf` file will be set by the default, BART format of the BART `archive_command` parameter since it is not explicitly set for this database server in the BART configuration file:

```
[BART]
bart-host= enterprisedb@192.168.2.22
backup_path = /opt/backup
pg_basebackup_path = /opt/PostgresPlus/9.4AS/bin/pg_basebackup
retention_policy = 6 BACKUPS
logfile = /tmp/bart.log
    .
    .
    .
[PG94_remote]
host = 192.168.2.24
port = 5432
user = postgres
backup-name = pg94_%year-%month-%dayT%hour:%minute
remote-host = postgres@192.168.2.24
copy_wals_during_restore = enabled
description = "PostgreSQL 94 remote server"
```

The default, BART `archive_command` format is the following:

```
archive_command = 'scp %p %h:%a/%f'
```

The `postgresql.auto.conf` file contains the following after the `INIT` subcommand is invoked:

```
# Do not edit this file manually!
# It will be overwritten by ALTER SYSTEM command.
archive_command = 'scp %p
enterprisedb@192.168.2.22:/opt/backup/pg94_remote/archived_wals/%f'
```

The `archive_command` uses the `scp` command since the BART backup catalog is remote relative to this database cluster. The BART user account, `enterprisedb`, is specified on the `scp` command since this is the user account owning the BART backup catalog where the archived WAL files are to be copied. The result is that there is no directory permission conflict during the archive operation.

6.5 BART Backup Catalog (backup_path)

Create the directory specified by the `backup_path` configuration parameter.

```
[BART]
bart-host= enterprisedb@192.168.2.22
backup_path = /opt/backup
pg_basebackup_path = /opt/PostgresPlus/9.4AS/bin/pg_basebackup
retention_policy = 6 BACKUPS
logfile = /tmp/bart.log
```

Make sure it is owned by the BART user account:

```
[root@localhost opt]# pwd
/opt
[root@localhost opt]# mkdir backup
[root@localhost opt]# chown enterprisedb backup
[root@localhost opt]# chgrp enterprisedb backup
[root@localhost opt]# chmod 700 backup
[root@localhost opt]# ls -l | grep backup
drwx------  2 enterprisedb enterprisedb 4096 Apr 23 15:36 backup
```

Use the BART `INIT` subcommand to complete the directory structure and set the Postgres `archive_command` configuration parameter for the 9.4 database servers.

Note: Before invoking any BART subcommands, set up a profile under the BART user account's home directory to set the `LD_LIBRARY_PATH` and `PATH` environment variables. See Step 3 in Section 4.1 for information.

Note: The `-o` option is specified with the `INIT` subcommand to force the setting of the Postgres `archive_command` configuration parameter for the 9.4 database servers when `archive_mode` is `off` or if the Postgres `archive_command` parameter is already set and needs to be overridden.

```
[user@localhost ~]$ su - enterprisedb
Password:
-bash-4.1$ bart INIT -o
```

```
INFO:  setting archive_command for server 'pg94_remote'
WARNING: archive_command is set. server restart is required
INFO:  setting archive_command for server 'ppas94'
WARNING: archive_command is set. server restart is required
```

The BART `SHOW-SERVERS` subcommand displays the following:

```
-bash-4.1$ bart SHOW-SERVERS
SERVER NAME         : pg94_remote
BACKUP FRIENDLY NAME: pg94_%year-%month-%dayT%hour:%minute
HOST NAME           : 192.168.2.24
USER NAME           : postgres
PORT                : 5432
REMOTE HOST         : postgres@192.168.2.24
RETENTION POLICY    : 6 Backups
DISK UTILIZATION    : 0.00 bytes
NUMBER OF ARCHIVES  : 0
ARCHIVE PATH        : /opt/backup/pg94_remote/archived_wals
ARCHIVE COMMAND     : (disabled)
XLOG METHOD         : fetch
WAL COMPRESSION     : disabled
TABLESPACE PATH(s)  :
DESCRIPTION         : "PostgreSQL 94 remote server"

SERVER NAME         : ppas93_remote
BACKUP FRIENDLY NAME: ppas93_%year-%month-%dayT%hour:%minute
HOST NAME           : 192.168.2.24
USER NAME           : repuser
PORT                : 5444
REMOTE HOST         : enterprisedb@192.168.2.24
RETENTION POLICY    : 6 Backups
DISK UTILIZATION    : 0.00 bytes
NUMBER OF ARCHIVES  : 0
ARCHIVE PATH        : /opt/backup/ppas93_remote/archived_wals
ARCHIVE COMMAND     : (disabled)
XLOG METHOD         : fetch
WAL COMPRESSION     : disabled
TABLESPACE PATH(s)  :
DESCRIPTION         : "PPAS 93 remote server"

SERVER NAME         : ppas94
BACKUP FRIENDLY NAME: ppas94_%year-%month-%dayT%hour:%minute
HOST NAME           : 127.0.0.1
USER NAME           : enterprisedb
PORT                : 5444
REMOTE HOST         :
RETENTION POLICY    : 6 Backups
DISK UTILIZATION    : 0.00 bytes
NUMBER OF ARCHIVES  : 0
ARCHIVE PATH        : /opt/backup/ppas94/archived_wals
ARCHIVE COMMAND     : (disabled)
XLOG METHOD         : fetch
WAL COMPRESSION     : disabled
TABLESPACE PATH(s)  :
DESCRIPTION         : "PPAS 94 server"

-bash-4.1$ cd /opt/backup
-bash-4.1$ pwd
/opt/backup
-bash-4.1$ ls -l
total 12
drwx------ 3 enterprisedb enterprisedb 4096 Apr 23 16:13 pg94_remote
```

```
drwx------ 3 enterprisedb enterprisedb 4096 Apr 23 16:13 ppas93_remote
drwx------ 3 enterprisedb enterprisedb 4096 Apr 23 16:13 ppas94
-bash-4.1$ cd pg94_remote
-bash-4.1$ ls -l
total 4
drwx------ 2 enterprisedb enterprisedb 4096 Apr 23 16:13 archived_wals
-bash-4.1$ cd ../ppas93_remote
-bash-4.1$ ls -l
total 4
drwx------ 2 enterprisedb enterprisedb 4096 Apr 23 16:13 archived_wals
-bash-4.1$ cd ../ppas94
-bash-4.1$ ls -l
total 4
drwx------ 2 enterprisedb enterprisedb 4096 Apr 23 16:13 archived_wals
```

Note: The `ARCHIVE PATH` field displays the full directory path to where the WAL files are copied. This directory path must match the directory path specified in the Postgres `archive_command` parameter of the `postgresql.conf` file or the `postgresql.auto.conf` file of each database server.

6.6 Start the Database Servers with WAL Archiving

After the BART backup catalog directory structure has been completed, begin the archiving of WAL files from the database servers by restarting each database server.

On BART host 192.168.2.22:

```
[root@localhost data]# /etc/init.d/ppas-9.4 restart
Restarting Postgres Plus Advanced Server 9.4:

WARNING --> PERL_INSTALL_PATH is not set in
/opt/PostgresPlus/9.4AS/etc/sysconfig/plLanguages.config file
WARNING --> PYTHON_INSTALL_PATH is not set in
/opt/PostgresPlus/9.4AS/etc/sysconfig/plLanguages.config file
WARNING --> TCL_INSTALL_PATH is not set in
/opt/PostgresPlus/9.4AS/etc/sysconfig/plLanguages.config file

waiting for server to shut down.... done
server stopped
waiting for server to start.... done
server started
Postgres Plus Advanced Server 9.4 restarted successfully
```

On remote host 192.168.2.24:

```
[root@localhost data]# /etc/init.d/ppas-9.3 restart
Restarting Postgres Plus Advanced Server 9.3:
waiting for server to shut down.... done
server stopped
waiting for server to start.... done
server started
Postgres Plus Advanced Server 9.3 restarted successfully

[root@localhost data]# /etc/init.d/postgresql-9.4 restart
Restarting PostgreSQL 9.4:
waiting for server to shut down.... done
server stopped
```

```
waiting for server to start.... done
server started
PostgreSQL 9.4 restarted successfully
```

In the BART backup catalog, if possible, verify that the WAL files are archiving.

Archived WAL files may not appear very frequently depending upon how often WAL archiving is set to switch to a new segment file in your database server configuration settings.

```
[root@localhost backup]# pwd
/opt/backup
[root@localhost backup]# ls -l
total 12
drwx------ 3 enterprisedb enterprisedb 4096 Apr 23 16:13 pg94 remote
drwx------ 3 enterprisedb enterprisedb 4096 Apr 23 16:13 ppas93_remote
drwx------ 3 enterprisedb enterprisedb 4096 Apr 23 16:13 ppas94
[root@localhost backup]# ls -l pg94_remote/archived_wals
total 16384
-rw------- 1 enterprisedb enterprisedb 16777216 Apr 23 16:40 00000001000000000000000B
[root@localhost backup]# ls -l ppas94/archived_wals
total 16384
-rw------- 1 enterprisedb enterprisedb 16777216 Apr 23 16:38 00000001000000000000000C
[root@localhost backup]# ls -l ppas93_remote/archived_wals
total 16384
-rw------- 1 enterprisedb enterprisedb 16777216 Apr 23 16:39 00000001000000000000000B
```

Also, verify that there are no archiving related errors in the database server log files.

6.7 Take a Base Backup

Take the first base backup of the database servers.

```
-bash-4.1$ bart BACKUP -s all -z

INFO:  creating backup for server 'pg94_remote'
INFO:  backup identifier: '1429821839345'
37589/37589 kB (100%), 1/1 tablespace

INFO:  backup completed successfully
INFO:  backup checksum: 6b61617171eee793de857b4ac4c5044b of base.tar.gz
INFO:
BACKUP DETAILS:
BACKUP STATUS: active
BACKUP IDENTIFIER: 1429821839345
BACKUP NAME: pg94_2015-04-23T16:43
BACKUP LOCATION: /opt/backup/pg94_remote/1429821839345
BACKUP SIZE: 2.45 MB
BACKUP FORMAT: tar.gz
XLOG METHOD: fetch
BACKUP CHECKSUM(s): 1
 ChkSum                            File
 6b61617171eee793de857b4ac4c5044b  base.tar.gz

TABLESPACE(s): 0
START WAL LOCATION: 00000001000000000000000C
BACKUP METHOD: streamed
BACKUP FROM: master
START TIME: 2015-04-23 16:43:59 EDT
STOP TIME: 2015-04-23 16:44:01 EDT
```

```
TOTAL DURATION: 2 sec(s)


INFO:  creating backup for server 'ppas93_remote'
INFO:  backup identifier: '1429821841418'
54645/54645 kB (100%), 1/1 tablespace

INFO:  backup completed successfully
INFO:  backup checksum: c6be1bcccf76050a7eea2fc502cf1796 of base.tar.gz
INFO:
BACKUP DETAILS:
BACKUP STATUS: active
BACKUP IDENTIFIER: 1429821841418
BACKUP NAME: pg94_2015-04-23T16:43
BACKUP LOCATION: /opt/backup/ppas93_remote/1429821841418
BACKUP SIZE: 5.37 MB
BACKUP FORMAT: tar.gz
XLOG METHOD: fetch
BACKUP CHECKSUM(s): 1
 ChkSum                            File
 c6be1bcccf76050a7eea2fc502cf1796   base.tar.gz

TABLESPACE(s): 0
START WAL LOCATION: 000000010000000000000000D
BACKUP METHOD: streamed
BACKUP FROM: master
START TIME: 2015-04-23 16:44:01 EDT
STOP TIME: 2015-04-23 16:44:03 EDT
TOTAL DURATION: 2 sec(s)


INFO:  creating backup for server 'ppas94'
INFO:  backup identifier: '1429821844271'
56356/56356 kB (100%), 1/1 tablespace

INFO:  backup completed successfully
INFO:  backup checksum: a1e4b6b2164ee59dfb0c3a6be9ee8ae3 of base.tar.gz
INFO:
BACKUP DETAILS:
BACKUP STATUS: active
BACKUP IDENTIFIER: 1429821844271
BACKUP NAME: pg94_2015-04-23T16:43
BACKUP LOCATION: /opt/backup/ppas94/1429821844271
BACKUP SIZE: 5.71 MB
BACKUP FORMAT: tar.gz
XLOG METHOD: fetch
BACKUP CHECKSUM(s): 1
 ChkSum                            File
 a1e4b6b2164ee59dfb0c3a6be9ee8ae3   base.tar.gz

TABLESPACE(s): 0
START WAL LOCATION: 000000010000000000000000E
STOP WAL LOCATION: 000000010000000000000000E
BACKUP METHOD: streamed
BACKUP FROM: master
START TIME: 2015-04-23 16:44:04 EDT
STOP TIME: 2015-04-23 16:44:06 EDT
TOTAL DURATION: 2 sec(s)

INFO:  all servers have been successfully backed up
```

The following shows the base backup directories created for each base backup of each database server. The base backup ID is used as the base backup directory name.

```
-bash-4.1$ cd /opt/backup
-bash-4.1$ pwd
/opt/backup
-bash-4.1$ ls -l
total 12
drwx------ 4 enterprisedb enterprisedb 4096 Apr 23 16:43 pg94_remote
drwx------ 4 enterprisedb enterprisedb 4096 Apr 23 16:44 ppas93_remote
drwx------ 4 enterprisedb enterprisedb 4096 Apr 23 16:44 ppas94
-bash-4.1$ ls -l pg94_remote
total 8
drwx------ 2 enterprisedb enterprisedb 4096 Apr 23 16:44 1429821839345
drwx------ 2 enterprisedb enterprisedb 4096 Apr 23 16:44 archived_wals
-bash-4.1$ ls -l ppas94
total 8
drwx------ 2 enterprisedb enterprisedb 4096 Apr 23 16:44 1429821844271
drwx------ 2 enterprisedb enterprisedb 4096 Apr 23 16:44 archived_wals
-bash-4.1$ ls -l ppas93_remote
total 8
drwx------ 2 enterprisedb enterprisedb 4096 Apr 23 16:44 1429821841418
drwx------ 2 enterprisedb enterprisedb 4096 Apr 23 16:44 archived_wals
```

6.8 Point-In-Time Recovery

The following demonstrates the point-in-time recovery operation on the remote PostgreSQL database server.

The following tables were created about one minute apart while WAL archiving is enabled:

```
postgres=# \dt
              List of relations
 Schema |       Name       | Type  |  Owner
--------+------------------+-------+----------
 public | pg94_rmt_t1_1650 | table | postgres
 public | pg94_rmt_t1_1652 | table | postgres
 public | pg94_rmt_t1_1654 | table | postgres
 public | pg94_rmt_t1_1700 | table | postgres
 public | pg94_rmt_t1_1701 | table | postgres
 public | pg94_rmt_t1_1702 | table | postgres
(6 rows)
```

In the table name `pg94_rmt_t`*`n`*`_`*`hhmi`*, *`n`* represents the active timeline. *`hhmi`* is the approximate time the table was created. For example, `pg94_rmt_t1_1650` was created at approximately 4:50 PM while timeline #1 is active.

The PostgreSQL database server was then stopped.

WAL files that have been created, but not yet archived must be identified, and then saved.

The following are the archived WAL files in the BART backup catalog:

```
-bash-4.1$ ls -l pg94 remote/archived wals
total 229380
-rw------- 1 enterprisedb enterprisedb 16777216 Apr 23 16:44 00000001000000000000000A
-rw------- 1 enterprisedb enterprisedb 16777216 Apr 23 16:40 00000001000000000000000B
-rw------- 1 enterprisedb enterprisedb 16777216 Apr 23 16:44 00000001000000000000000C
-rw------- 1 enterprisedb enterprisedb      302 Apr 23 16:44
00000001000000000000000C.00000028.backup
-rw------- 1 enterprisedb enterprisedb 16777216 Apr 23 16:49 00000001000000000000000D
-rw------- 1 enterprisedb enterprisedb 16777216 Apr 23 16:50 00000001000000000000000E
-rw------- 1 enterprisedb enterprisedb 16777216 Apr 23 16:51 00000001000000000000000F
-rw------- 1 enterprisedb enterprisedb 16777216 Apr 23 16:52 000000010000000000000010
-rw------- 1 enterprisedb enterprisedb 16777216 Apr 23 16:53 000000010000000000000011
-rw------- 1 enterprisedb enterprisedb 16777216 Apr 23 16:54 000000010000000000000012
-rw------- 1 enterprisedb enterprisedb 16777216 Apr 23 16:55 000000010000000000000013
-rw------- 1 enterprisedb enterprisedb 16777216 Apr 23 16:59 000000010000000000000014
-rw------- 1 enterprisedb enterprisedb 16777216 Apr 23 17:00 000000010000000000000015
-rw------- 1 enterprisedb enterprisedb 16777216 Apr 23 17:01 000000010000000000000016
-rw------- 1 enterprisedb enterprisedb 16777216 Apr 23 17:02 000000010000000000000017
```

The following lists the current PostgreSQL server WAL files. The ones that have not been archived are indicated with bold font.

```
-bash-4.1$ cd /opt/PostgreSQL/9.4/data/pg_xlog
-bash-4.1$ pwd
/opt/PostgreSQL/9.4/data/pg_xlog
-bash-4.1$ ls -l
total 114696
-rw------- 1 postgres postgres      302 Apr 23 16:44
00000001000000000000000C.00000028.backup
-rw------- 1 postgres postgres 16777216 Apr 23 17:02 000000010000000000000017
-rw------- 1 postgres postgres 16777216 Apr 23 17:02 000000010000000000000018
-rw------- 1 postgres postgres 16777216 Apr 23 16:54 000000010000000000000019
-rw------- 1 postgres postgres 16777216 Apr 23 16:55 00000001000000000000001A
-rw------- 1 postgres postgres 16777216 Apr 23 17:00 00000001000000000000001B
-rw------- 1 postgres postgres 16777216 Apr 23 16:59 00000001000000000000001C
-rw------- 1 postgres postgres 16777216 Apr 23 17:01 00000001000000000000001D
drwx------ 2 postgres postgres     4096 Apr 23 17:02 archive_status
```

Copies of the unarchived WAL files are saved to a temporary location:

```
-bash-4.1$ mkdir /tmp/unarchived pg94 wals
-bash-4.1$ pwd
/opt/PostgreSQL/9.4/data/pg_xlog
-bash-4.1$ cp -p 000000010000000000000018 /tmp/unarchived_pg94_wals
-bash-4.1$ cp -p 000000010000000000000019 /tmp/unarchived pg94 wals
-bash-4.1$ cp -p 00000001000000000000001A /tmp/unarchived_pg94_wals
-bash-4.1$ cp -p 00000001000000000000001B /tmp/unarchived pg94 wals
-bash-4.1$ cp -p 00000001000000000000001C /tmp/unarchived_pg94_wals
-bash-4.1$ cp -p 00000001000000000000001D /tmp/unarchived_pg94_wals
-bash-4.1$ ls -l /tmp/unarchived_pg94_wals
total 98304
-rw------- 1 postgres postgres 16777216 Apr 23 17:02 000000010000000000000018
-rw------- 1 postgres postgres 16777216 Apr 23 16:54 000000010000000000000019
-rw------- 1 postgres postgres 16777216 Apr 23 16:55 00000001000000000000001A
-rw------- 1 postgres postgres 16777216 Apr 23 17:00 00000001000000000000001B
-rw------- 1 postgres postgres 16777216 Apr 23 16:59 00000001000000000000001C
-rw------- 1 postgres postgres 16777216 Apr 23 17:01 00000001000000000000001D
```

On the remote host, the directory is created to which the PostgreSQL database cluster is to be restored. This restore path is `/opt/restore_pg94` owned by user account `postgres`.

```
[user@localhost ~]$ su root
Password:
[root@localhost user]# cd /opt
[root@localhost opt]# mkdir restore_pg94
[root@localhost opt]# chown postgres restore_pg94
[root@localhost opt]# chgrp postgres restore_pg94
[root@localhost opt]# chmod 700 restore_pg94
[root@localhost opt]# ls -l
total 16
drwxr-xr-x  3 root     daemon   4096 Apr 23 12:56 PostgresPlus
drwxr-xr-x  3 root     daemon   4096 Apr 23 12:59 PostgreSQL
drwx------  2 postgres postgres 4096 Apr 23 17:23 restore_pg94
drwxr-xr-x. 2 root     root     4096 Nov 22  2013 rh
```

Note that in the BART configuration file, the remote user and remote host IP address, `postgres@192.168.2.24`, have been set with the `remote-host` parameter. If not given in the BART configuration file, this information must then be specified by the `--remote-host` option when giving the `RESTORE` subcommand (for example, `bart RESTORE --remote-host postgres@192.168.2.24 ...`).

```
[PG94_remote]
host = 192.168.2.24
port = 5432
user = postgres
backup-name = pg94_%year-%month-%dayT%hour:%minute
remote-host = postgres@192.168.2.24
copy_wals_during_restore = enabled
description = "PostgreSQL 94 remote server"
```

Use the `SHOW-BACKUPS` subcommand to identify the base backup to use with the `RESTORE` subcommand.

```
-bash-4.1$ bart SHOW-BACKUPS
 SERVER NAME     BACKUP ID       BACKUP TIME               BACKUP SIZE   WAL(s) SIZE
WAL FILES   STATUS

 pg94_remote     1429821839345   2015-04-23 16:44:01 EDT   2.45 MB       192.00 MB
12          active
 ppas93_remote   1429821841418   2015-04-23 16:44:03 EDT   5.38 MB       32.00 MB
2           active
 ppas94          1429821844271   2015-04-23 16:44:07 EDT   5.71 MB       16.00 MB
1           active
```

The `-t` option with the `SHOW-BACKUPS` subcommand also displays the user-defined backup name if one was created:

```
-bash-4.1$ bart SHOW-BACKUPS -s pg94_remote -i 1429821839345 -t
SERVER NAME    : pg94_remote
BACKUP ID      : 1429821839345
BACKUP NAME    : pg94_2015-04-23T16:43
BACKUP STATUS  : active
BACKUP TIME    : 2015-04-23 16:44:01 EDT
BACKUP SIZE    : 2.45 MB
WAL(S) SIZE    : 192.00 MB
NO. OF WALS    : 12
FIRST WAL FILE : 000000010000000000000000C
CREATION TIME  : 2015-04-23 16:44:06 EDT
LAST WAL FILE  : 000000010000000000000017
```

```
CREATION TIME  : 2015-04-23 17:02:22 EDT
```

A recovery is made using timeline 1 to 2015-04-23 16:57:00.

```
-bash-4.1$ bart RESTORE -s pg94_remote -i pg94_2015-04-23T16:43 -p
/opt/restore_pg94 -t 1 -g '2015-04-23 16:57:00'
INFO:  restoring backup 'pg94_2015-04-23T16:43' of server 'pg94_remote'
INFO:  restoring backup to postgres@192.168.2.24:/opt/restore_pg94
INFO:  base backup restored
INFO:  copying WAL file(s) to
postgres@192.168.2.24:/opt/restore_pg94/archived_wals
INFO:  creating recovery.conf file
INFO:  archiving is disabled
```

The following shows the restored base backup files in the restore path directory, `/opt/restore_pg94`:

```
-bash-4.1$ pwd
/opt/restore_pg94
-bash-4.1$ ls -l
total 124
drwxr-xr-x 2 postgres postgres  4096 Apr 23 17:44 archived_wals
-rw------- 1 postgres postgres   206 Apr 23 16:43 backup_label
drwx------ 5 postgres postgres  4096 Apr 23 12:59 base
drwx------ 2 postgres postgres  4096 Apr 23 17:44 global
drwx------ 2 postgres postgres  4096 Apr 23 12:59 pg_clog
drwx------ 2 postgres postgres  4096 Apr 23 12:59 pg_dynshmem
-rw------- 1 postgres postgres  4212 Apr 23 14:14 pg_hba.conf
-rw------- 1 postgres postgres  1636 Apr 23 12:59 pg_ident.conf
drwxr-xr-x 2 postgres postgres  4096 Apr 23 16:43 pg_log
drwx------ 4 postgres postgres  4096 Apr 23 12:59 pg_logical
drwx------ 4 postgres postgres  4096 Apr 23 12:59 pg_multixact
drwx------ 2 postgres postgres  4096 Apr 23 16:39 pg_notify
drwx------ 2 postgres postgres  4096 Apr 23 12:59 pg_replslot
drwx------ 2 postgres postgres  4096 Apr 23 12:59 pg_serial
drwx------ 2 postgres postgres  4096 Apr 23 12:59 pg_snapshots
drwx------ 2 postgres postgres  4096 Apr 23 16:39 pg_stat
drwx------ 2 postgres postgres  4096 Apr 23 16:43 pg_stat_tmp
drwx------ 2 postgres postgres  4096 Apr 23 12:59 pg_subtrans
drwx------ 2 postgres postgres  4096 Apr 23 12:59 pg_tblspc
drwx------ 2 postgres postgres  4096 Apr 23 12:59 pg_twophase
-rw------- 1 postgres postgres     4 Apr 23 12:59 PG_VERSION
drwx------ 3 postgres postgres  4096 Apr 23 17:44 pg_xlog
-rw------- 1 postgres postgres   178 Apr 23 16:13 postgresql.auto.conf
-rw-r--r-- 1 postgres postgres 21279 Apr 23 17:44 postgresql.conf
-rw-r--r-- 1 postgres postgres   118 Apr 23 17:44 recovery.conf
```

Copy the saved, unarchived WAL files to the restore path `pg_xlog` subdirectory (`/opt/restore_pg94/pg_xlog`):

```
-bash-4.1$ pwd
/opt/restore pg94/pg xlog
-bash-4.1$ ls -l
total 16388
-rw------- 1 postgres postgres 16777216 Apr 23 16:44 000000010000000000000000C
drwx------ 2 postgres postgres     4096 Apr 23 17:44 archive_status
-bash-4.1$ ls -l /tmp/unarchived_pg94_wals
total 98304
-rw------- 1 postgres postgres 16777216 Apr 23 17:02 000000010000000000000018
-rw------- 1 postgres postgres 16777216 Apr 23 16:54 000000010000000000000019
```

```
-rw------- 1 postgres postgres 16777216 Apr 23 16:55 00000001000000000000001A
-rw------- 1 postgres postgres 16777216 Apr 23 17:00 00000001000000000000001B
-rw------- 1 postgres postgres 16777216 Apr 23 16:59 00000001000000000000001C
-rw------- 1 postgres postgres 16777216 Apr 23 17:01 00000001000000000000001D
-bash-4.1$ cp -p /tmp/unarchived_pg94_wals/* .
-bash-4.1$ ls -l
total 114692
-rw------- 1 postgres postgres 16777216 Apr 23 16:44 00000001000000000000000C
-rw------- 1 postgres postgres 16777216 Apr 23 17:02 000000010000000000000018
-rw------- 1 postgres postgres 16777216 Apr 23 16:54 000000010000000000000019
-rw------- 1 postgres postgres 16777216 Apr 23 16:55 00000001000000000000001A
-rw------- 1 postgres postgres 16777216 Apr 23 17:00 00000001000000000000001B
-rw------- 1 postgres postgres 16777216 Apr 23 16:59 00000001000000000000001C
-rw------- 1 postgres postgres 16777216 Apr 23 17:01 00000001000000000000001D
drwx------ 2 postgres postgres     4096 Apr 23 17:44 archive_status
```

Inspect the `/opt/restore_pg94/recovery.conf` file to verify that it contains the correct recovery settings:

```
restore_command = 'cp archived_wals/%f %p'
recovery_target_time = '2015-04-23 16:57:00'
recovery_target_timeline = 1
```

Note that it restores from the `archived_wals` subdirectory of `/opt/restore_pg94` since the `copy_wals_during_restore` parameter in the BART configuration file is set to `enabled` for database server `PG94_remote`.

Start the database server to initiate the point-in-time recovery operation.

```
[user@localhost ~]$ su postgres
Password:
bash-4.1$ cd /opt/restore_pg94
bash-4.1$ /opt/PostgreSQL/9.4/bin/pg_ctl start -D /opt/restore_pg94 -l
/opt/restore_pg94/pg_log/logfile
server starting
```

Inspect the database server log file to ensure the operation did not result in any errors.

```
2015-04-23 18:01:08 EDT LOG:  database system was interrupted; last known up at 2015-
04-23 16:43:59 EDT
2015-04-23 18:01:08 EDT LOG:  starting point-in-time recovery to 2015-04-23 16:57:00-04
2015-04-23 18:01:08 EDT LOG:  restored log file "00000001000000000000000C" from archive
2015-04-23 18:01:08 EDT LOG:  redo starts at 0/C000090
2015-04-23 18:01:08 EDT LOG:  consistent recovery state reached at 0/C0000B8
2015-04-23 18:01:08 EDT LOG:  restored log file "00000001000000000000000D" from archive
2015-04-23 18:01:08 EDT LOG:  restored log file "00000001000000000000000E" from archive
2015-04-23 18:01:08 EDT LOG:  restored log file "00000001000000000000000F" from archive
2015-04-23 18:01:08 EDT LOG:  restored log file "000000010000000000000010" from archive
2015-04-23 18:01:08 EDT LOG:  restored log file "000000010000000000000011" from archive
2015-04-23 18:01:08 EDT LOG:  restored log file "000000010000000000000012" from archive
2015-04-23 18:01:08 EDT LOG:  restored log file "000000010000000000000013" from archive
2015-04-23 18:01:08 EDT LOG:  restored log file "000000010000000000000014" from archive
2015-04-23 18:01:08 EDT LOG:  restored log file "000000010000000000000015" from archive
2015-04-23 18:01:08 EDT LOG:  recovery stopping before commit of transaction 1815, time
2015-04-23 17:00:24.475528-04
2015-04-23 18:01:08 EDT LOG:  redo done at 0/150186E0
2015-04-23 18:01:08 EDT LOG:  last completed transaction was at log time 2015-04-23
16:54:09.212224-04
cp: cannot stat `archived_wals/00000002.history': No such file or directory
2015-04-23 18:01:08 EDT LOG:  selected new timeline ID: 2
cp: cannot stat `archived_wals/00000001.history': No such file or directory
2015-04-23 18:01:08 EDT LOG:  archive recovery complete
```



```
2015-04-23 18:01:08 EDT LOG:  database system is ready to accept connections
2015-04-23 18:01:08 EDT LOG:  autovacuum launcher started
```

The tables that exist in the recovered database cluster are the following:

```
postgres=# \dt
              List of relations
 Schema |       Name       | Type  |  Owner
--------+------------------+-------+----------
 public | pg94_rmt_t1_1650 | table | postgres
 public | pg94_rmt_t1_1652 | table | postgres
 public | pg94_rmt_t1_1654 | table | postgres
(3 rows)
```

Since recovery was up to and including 2015-04-23 16:57:00, the following tables created after 16:57 are not present:

```
 public | pg94_rmt_t1_1700 | table | postgres
 public | pg94_rmt_t1_1701 | table | postgres
 public | pg94_rmt_t1_1702 | table | postgres
```

Note: The BART `RESTORE` operation stops WAL archiving by adding an `archive_mode = off` parameter at the very end of the `postgresql.conf` file. This last parameter in the file overrides any other previous setting of the same parameter in the file. Delete the last setting and restart the database server to start WAL archiving.

```
# Add settings for extensions here
archive_mode = off
```

7 Known Issues

This chapter contains a list of known issues for BART. (The number enclosed in parentheses at the end of each issue is for internal use only.)

1. For the BART `RESTORE` subcommand, if invalid values are specified for the options, or if invalid combinations are specified (for example, if both the `-x` *`target_xid`* and the `-g` *`target_timestamp`* options are given), no error message is generated by BART. The invalid options are accepted and passed to the `recovery.conf` file, which is then processed by the Postgres database server. Be sure that valid options are specified when using the `RESTORE` subcommand. (33113)